applied sciences

Special Issue Reprint

Applied Machine Learning

Edited by
Grzegorz Dudek

www.mdpi.com/journal/applsci

MDPI

Applied Machine Learning

Applied Machine Learning

Editor

Grzegorz Dudek

MDPI • Basel • Beijing • Wuhan • Barcelona • Belgrade • Manchester • Tokyo • Cluj • Tianjin

Editor
Grzegorz Dudek
Czestochowa University of Technology
Poland

Editorial Office
MDPI
St. Alban-Anlage 66
4052 Basel, Switzerland

This is a reprint of articles from the Special Issue published online in the open access journal *Applied Sciences* (ISSN 2076-3417) (available at: https://www.mdpi.com/journal/applsci/special_issues/Applied_Machine_Learning).

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

LastName, A.A.; LastName, B.B.; LastName, C.C. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range.

ISBN 978-3-0365-7906-1 (Hbk)
ISBN 978-3-0365-7907-8 (PDF)

Contents

About the Editor

Grzegorz Dudek

Grzegorz Dudek received his PhD in electrical engineering from Czestochowa University of Technology (CUT), Poland, in 2003 and his habilitation in computer science from Lodz University of Technology, Poland, in 2013. Currently, he is an associate professor at the Department of Electrical Engineering, CUT. He is the author of four books on the subject of machine learning for forecasting and evolutionary algorithms for unit commitment in addition to over 100 scientific papers. He came third in the Global Energy Forecasting Competition 2014—price forecasting track. His research interests include machine learning and artificial intelligence, and their application to practical classification, regression, forecasting and optimization problems.

applied sciences

MDPI

Editorial

Special Issue on Applied Machine Learning

Grzegorz Dudek

Department of Electrical Engineering, Częstochowa University of Technology, 42-200 Częstochowa, Poland; grzegorz.dudek@pcz.pl

1. Introduction

Machine learning (ML) is one of the most exciting fields of computing today. Over the past few decades, ML has become an entrenched part of everyday life and has been successfully used to solve practical problems. An application area of ML is very broad, including engineering, industry, business, finance, medicine, and many other domains. ML covers a wide range of learning algorithms, including classical ones such as linear regression, k-nearest neighbors, decision trees, support vector machines and neural networks, and newly developed algorithms such as deep learning and boosted tree models. In practice, it is quite challenging to properly determine an appropriate architecture and parameters of ML models so that the resulting learner model can achieve sound performance for both learning and generalization. Practical applications of ML bring additional challenges, such as dealing with big, missing, distorted, and uncertain data. In addition, interpretability is a paramount quality that ML methods should aim to achieve if they are to be applied in practice. Interpretability allows us to understand ML model operation and raises confidence in its results.

This book compiles 41 papers published in the Special Issue titled “Applied Machine Learning”. The papers focus on applications of ML models in a diverse range of fields and problems. They report substantive results on a wide range of learning methods, discuss conceptualization of problems, data representation, feature engineering, ML models, critical comparisons with existing techniques, and interpretation of results.

2. Summary of the Contributions

There were 116 papers submitted to this special issue, and 41 papers were accepted. Although each paper covers different topics, we can identify six categories where the papers can be classified according to their main focus: computer vision, teaching and learning, social media, forecasting, basic problems of ML, and other topics.

2.1. Computer Vision

Image processing and analysis are a basis of computer vision problems such as semantic segmentation, object classification, localization, and detection, optical character recognition, facial recognition etc. An appropriate representation of image content is a crucial problem. In [1], to deal with this problem, a novel type of representation is proposed where an image is reduced to a set of highly sparse matrices representing detected keypoints. The authors express intensity of features extracted from a dedicated convolutional neural network (CNN) autoencoder. The new features have many advantages such as they are not manually designed but learned, they are expected to minimize information loss and they are relatively easy to interpret.

In [2], a fast-self-adaptive digital camouflage method based on deep learning is proposed. It is designed for the new generation of adaptive optical camouflage which can change with the environment in real-time. The system is composed of a YOLOv3 model that identifies military targets, a pre-trained deepfillv1 model that designs the preliminary camouflage texture, and a k-means algorithm for standardization of the texture.

Citation: Dudek, G. Special Issue on Applied Machine Learning. *Appl. Sci.* **2022**, *12*, 2039. https://doi.org/10.3390/app12042039

Received: 13 January 2022
Accepted: 13 February 2022
Published: 16 February 2022

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

The experimental results show that the camouflage pattern designed by the proposed method is consistent with the background in texture and semantics, and has excellent camouflage performance.

A problem of classification of remote sensing images for disaster investigation, traffic control, and land-use resource management is considered in [3]. A new remote sensing scene classification network is proposed and a two-stage cyclical learning is developed to speed up model training and enhance accuracy. A t-distributed stochastic neighbor embedding algorithm was used to verify the effectiveness of the proposed model, and a local interpretable model-agnostic explanation algorithm was applied to improve the results.

In [4], tracking pedestrian workers on construction sites is considered to improve efficiency and safety management. Vision-based tracking approaches, suitable in this case, require a large amount of data originating from construction sites. These data are hardly available, so the authors propose to use a small general dataset and combine a deep learning detector with an approach based on classical ML techniques. They use YOLOv3 detector for identifying workers and compare its performance with an approach based on a soft cascaded classifier. They found that both approaches generally yield satisfying tracking performances but feature different characteristics. To augment a self-recorded real world dataset for learning the vision-based tracking system, in [5] virtual construction site scenarios are modeled using 3D computer graphics software. The detector's performance is examined when using synthetic data of various environmental conditions for training. The findings showed that a synthetic extension is beneficial for otherwise small datasets. It is an alternative to evaluate vision-based tracking systems on hazardous scenes without exposing workers to risks.

In [6], a problem of environment classification for unmanned aerial vehicles (UAV) is addressed. Images obtained from video and photographic cameras mounted on a UAV are recognized to detect ground, sky, and clouds. The proposed recognition system includes CNN trained with a dataset generated by both, a human expert and a Support Vector Machine (SVM) to capture context and precise localization.

2.2. Teaching and Learning

Student grade prediction is an important educational problem for designing personalized strategies of teaching and learning. To solve this problem, in [7], a graph regularized robust matrix factorization is proposed optimized by a majorization minimization algorithm. This method integrates two side graphs built on the side data of students and courses into the objective of robust low-rank matrix factorization. As a result, the learned features of students and courses can grasp more priors from educational situations to achieve higher grade prediction results. This facilitates personalized teaching and learning in higher education.

For developing adaptive e-learning systems, it is very helpful to provide information on how students recognize, process and store information. To improve students' learning evaluation, in [8], a method based on deep multi-target prediction algorithm using Felder-Silverman learning styles model is proposed. It uses feature selection, learning styles models, and multiple target classification to investigate the possibility of improving the accuracy of automatic learning styles identification. The obtained results show that learning styles allow adaptive e-learning systems to improve the learning processes of students.

Students' performance prediction in higher education was considered in [9]. To exploit the knowledge retrieved from one problem for improving the predictive performance of a learning model for a different but related problem, the authors use transfer learning. The experimental results demonstrate that the prognosis of students at risk of failure can be achieved with satisfactory accuracy in most cases, provided that datasets of students who have attended other related courses are available.

Paper [10] was conducted with the aim of identifying the interrelationships among topics based on the understanding of various bodies of knowledge. The study provides a foundation for topic compositions to construct an academic body of knowledge of AI.

To this end, ML-based sentence similarity measurement models used in machine translation, chatbots, and document summarization were applied to the body of knowledge of AI. Consequently, several similar topics related to agent designing in AI were identified. The results of this study can be applied in the edutech field.

Predicting the academic standing of a student at the graduation time can be very useful for institutions to select among candidates or in helping potentially weak students in overcoming educational challenges. In [11], this problem is solved using several ML algorithms based on different student data including individual course grades and grade point averages. This approach can be applied to any dataset to determine when to use which college performance representation for enhanced prediction. For predicting the grades of undergraduate students in the final exams, in [12], multi-view learning is applied to exploit the knowledge retrieved from data, represented by multiple feature subsets known as views. A semi-supervised regression algorithm is proposed which exploits three independent and naturally formed feature views, derived from different sources. The experimental results demonstrate that the early prognosis of students at risk of failure can be accurately achieved and could highly benefit the educational domain.

2.3. Social Media

One prominent dark side of online information behavior is the spreading of rumors on social media. Paper [13] analyses the association between user features and rumor refuting behavior in different rumor categories. Natural language processing (NLP) techniques are applied to quantify the user's sentiment tendency and recent interests. The users' personalized features are used to train XGBoost classification model to identify potential refuters. The results revealed that there are significant differences between rumor stiflers and refuters, as well as between refuters for different categories.

The objective of [14] is to detect variables that allow organizations to manage their social network services efficiently. This study, applying ML algorithms and multiple linear regression, reveals which aspects of published content increase the recognition of publications through retweets and favorites. The findings of this research provide new knowledge about trends and patterns of use in social media, providing academics and professionals with the necessary guidelines to efficiently manage these technologies in the organizational field.

Paper [15] concerns the tourists' sentiments regarding travel destinations based on online travel review texts. The authors transformed sentiment analysis into a multi-classification problem based on ML methods, and further designed a keyword semantic expansion method based on a knowledge graph. The method extracts keywords from online travel review texts and obtains the concept list of keywords through the knowledge graph. This list is then added to the review text to facilitate the construction of semantically expanded classification data. The results of sentiment analysis form an important basis for tourism decision making.

Micro-blogs, such as Twitter, have become important tools to share opinions and information among users. The authors of [16] wonder how a user can discover influencers concerned with their interest. They propose a classification model trained on messages labeled with topical classes, so as this model is able to classify unlabeled messages. This model can be used to reveal the hidden topic the messages are talking about.

With the widespread use of over-the-top (OTT) media, such as YouTube and Netflix, network markets are changing and innovating rapidly, making it essential for network providers to quickly and efficiently analyze OTT traffic with respect to pricing plans and infrastructure investments. In [17], a time-aware deep learning method of analyzing OTT traffic to classify users for this purpose is presented. A novel framework to better exploit accuracy, while dramatically reducing classification time is proposed. The resultant approach provides a simple method for customizing pricing plans and load balancing by classifying OTT users more accurately.

Recommendation systems aim to decipher user interests, preferences, and behavioral patterns automatically. The credibility of the recommendation is of magnificent importance in crowdfunding project recommendations. Paper [18] devises a hybrid ML-based approach for credible crowdfunding projects' recommendations by wisely incorporating backers' sentiments and other influential features. The proposed model has four modules: a feature extraction module, a hybrid latent Dirichlet allocation and LSTM-based latent topics evaluation module, credibility formulation, and recommendation module. The proposed model's evaluation depicts that credibility assessment based on the hybrid ML approach contributes more efficient results than existing recommendation models.

2.4. Forecasting

Stock performance prediction is one of the most challenging issues in time series data analysis. Paper [19] proposes to build an automated trading system by integrating AI and the proven method invented by human stock traders. The knowledge and experience of the successful stock traders are extracted from their related publications. After that, an LSTM-based deep NN is developed to use the human stock traders' knowledge in the automatic trading system. Experimental results indicate that the proposed ranking-based stock classification considering historical volatility strategy outperforms conventional methods.

In [20], the authors study the volatility forecasts in the Bitcoin market, which has become popular in the global market in recent years. For the improvement of the forecasting accuracy of Bitcoin's volatility, they develop hybrid forecasting models combining the GARCH family models with the ML approach including NNs.

Paper [21] is about forecasting the Key Performance Indicators (KPIs), usually in the form of time series data, related to the COVID-19 pandemic. Making reliable predictions of these indicators, particularly for emergency departments, can facilitate acute unit planning, enhance the quality of care and optimise resources. The authors compare the KPI forecasting models including classical ARIMA, Prophet and General Regression NN.

A development of the intelligent transport systems has created conditions for solving the supply-demand imbalance of public transportation services. In [22], a method to forecast real-time online taxi-hailing demand is introduced. It is based on NNs and extreme gradient boosting. The proposed method can help to schedule online taxi-hailing resources in advance.

Climate change increases the frequency and intensity of heatwaves, causing significant human and material losses every year. Big data, whose volumes are rapidly increasing, are expected to be used for preemptive responses. In [23], for weekly prediction of heat-related damages, a random forest model was developed using statistical, meteorological, and floating population data. The results show that the proposed model outperforms existing ones.

One of the hottest topics in today's meteorological research is weather nowcasting, which is the weather forecast for a short time period such as one to six hours. With the main goal of helping meteorologists in analyzing radar data for issuing nowcasting warnings, in [24], a regression model based on an ensemble of deep NNs for predicting the values for radar products is proposed. The proposed model is intended to be a proof of concept for the effectiveness of learning from radar data relevant patterns that would be useful for predicting future values for radar products based on their historical values.

2.5. Basic Problems of ML

Paper [25] deals with the problem of instance selection for classifiers. The main goal is to improve the performance of a classifier (its speed and accuracy) by eliminating redundant and noisy samples. The obtained results indicate that for the most of the classifiers compressing the training set affects prediction performance and only a small group of instance selection methods can be recommended as a general purpose preprocessing step. These are learning vector quantization based algorithms, along with the Drop2 and Drop3.

Support vector machines are a well-known classifiers due to their superior classification performance. To decrease the large-scale SVM complexity, in [26], a novel data reduction method for reducing the training time by combining decision trees and relative support distance is proposed. The method selects good support vector candidates in each partition generated by the decision trees. The selected candidates reduced the training time while maintaining good classification performance in comparison with existing approaches

Paper [27] deals with the problem of solving of partial differential equations, which is a hot topic of mathematical research. The authors introduce an improved Physics Informed Neural Network (PINN) for solving partial differential equations. PINN takes the physical information that is contained in partial differential equations as a regularization term, which improves a performance of NNs. The experimental results show that PINN is effective in solving partial differential equations and deserves further research.

Machine learning of automata and grammars has a wide range of applications in such fields as syntactic pattern recognition, computational biology, systems modeling, natural language acquisition, and knowledge discovery. In [28], an approach to non-deterministic finite automaton inductive synthesis that is based on answer set programming (ASP) solvers are proposed. They consist of preparing logical rules before starting the searching process. The authors show how the proposed ASP solvers help to tackle the regular inference problem for large-size instances and compare their approach with the existing ones. Experiments indicated that the proposed approach clearly outperforms the current state-of-the-art satisfiability-based method and all backtracking algorithms proposed in the literature.

Paper [29] sits in the scientific field known as grammatical inference (GI), automata learning, grammar identification, or grammar induction. The matter under consideration is the set of rules that lie behind a given sequence of words and the main task is to discover the rules that can help to evaluate new, unseen words. The authors propose a new grammatical inference method and applied it to a real bioinformatics task, i.e., classification of amyloidogenic sequences. In the experimental evaluation, they showed that the new grammatical inference algorithm gives the best results in comparison to other automata or grammar learning methods as well as ML approach combining an unsupervised data-driven distributed representation and SVM.

Paper [30] is about anticipatory classifier systems, i.e., the classifier systems that learn by using a cognitive mechanism of anticipatory behavioral control which was introduced in cognitive psychology. The authors note that the learning classifier systems revealed many real-world sequential decision problems where the preferred objective is the maximization of the average of successive rewards. To address such problems, they proposes a modification toward a learning component: a new average reward criterion. In the experimental study, they showed that the anticipatory classifier systems with an averaged reward criterion can be used successively in multi-step environments.

2.6. Other Topics

A medical care application of ML is considered in [31]. This work is on a sleep apnea which is a common sleep-related disorder that significantly affects the population. It is characterized by repeated breathing interruption during sleep. The authors propose a new probabilistic algorithm based on oronasal respiration signal for automated detection of apnea events during sleep. Unlike classical threshold-based classification models, they use a Gaussian mixture probability model for detecting sleep apnea based on the posterior probabilities of the respective events. The results show significant improvement in the ability to detect sleep apnea events compared to a rule-based classifier that uses the same classification features and also compared to the previously published studies.

Paper [32] deals with a discrete optimization problem of product placement and of order picking routes in a warehouse. The authors propose a genetic algorithm that minimizes the sum of the order picking times. The product placement is optimized by another genetic algorithm. To improve and accelerate an optimization process, several ideas are proposed such as a multi-parent crossover, caching procedure, multiple restart and order grouping. A proposed solution decreases significantly the total order picking times.

ML techniques have been actively applied to the meteorology and climatology fields in recent years. They are used for forecasting in different horizons, modelling climatic data, quality control and correction of observed weather data. Paper [33] deals with the topic of the climate change. It presents a framework for selecting general circulation models (GCMs) in homogeneous climatic zones and detecting future climate change trends. With the support of ML techniques, long records of climate data, from numerous gauging sites and web sources, were analyzed and used to determine historical and projected trends of climate change. In [34], to detect the weather phenomena such as precipitation and fog from the backscatter data obtained from the lidar ceilometer, three ML models were applied: random forest, SVM, and NN. The prediction results showed the potential for precipitation detection, but fog detection was found to be very difficult.

The emission of carbon dioxide caused by various sectors, including construction and industrial processes, has emerged as a severe problem that dramatically affects global climate change. A portland cement production process accounts for a large part global anthropogenic CO_2 emission. A fly ash-based geopolymer concrete (FAGP) offers a favourable alternative to conventional Portland concrete due to its reduced embodied carbon dioxide content. In [35], ML methods including artificial NN, deep NN and ResNet were employed to predict mechanical properties of FAGP concrete. The obtained results indicate that the proposed approaches offer reliable methods for FAGP design and optimisation.

Paper [36] deals with a vibration test in the space structure testing. During the physical tests, the structure must not be overtested to avoid any risk of damage. In order to solve the issues associated with existing methods of live monitoring of the structure's response, the authors investigated the use of artificial NNs to predict the system's responses during the test. The conducted research accounts for a novel method for live prediction of stresses, allowing failure to be evaluated for different types of material via yield criteria.

Software vulnerabilities are one of the main causes of cybersecurity problems, resulting in huge losses. Existing solutions to automated vulnerability detection are mostly based on features that are defined by human experts and directly lead to missed potential vulnerability. Deep learning is proposed in [37] as an effective method for automating the extraction of vulnerability characteristics. Word2vec continuous bag-of-words, multiple structural CNNs, and stacking classifiers were found to be the best combination for automated vulnerability detection by comparing classification results.

Paper [38] is on information privacy which is a critical design feature for any exchange system, with privacy-preserving applications requiring the identification and labelling of sensitive information. The authors propose a predictive context-aware model based on a Bidirectional Long Short Term Memory network with Conditional Random Fields (BiLSTM + CRF) to identify and label sensitive information in conversational data. The results demonstrate that the BiLSTM + CRF model architecture with BERT embeddings and WordShape features is very effective and outperforms competitive solutions.

Natural language processing has enormous areas of applications including sentiment analysis, machine translation, text classification and extraction. In [39], the problem of developing a deep learning-based language model that helps software engineers write code faster is considered. This research proposes a hybrid approach that harnesses the synergy between ML techniques and advanced design methods aiming to develop a code auto-completion framework that helps firmware developers write code in a more efficient manner. The proposed framework can save numerous hours of productivity by eliminating tedious parts of writing code.

In [40], the problem of predicting the movement of a drifter on the ocean is considered. The authors estimated drifter tracking over seawater using ML and evolutionary search techniques including differential evolution, particle swarm optimization, multi-layer perceptron, SVM, deep NNs, LSTM and others. Extensive comparative research allows us to evaluate the suitability of various ML algorithms for solving this type of problem.

A salesperson performance measurement is a process that occurs multiple times per year on a company. During this process, the salesperson is evaluated how he or she performed on numerous KPIs. In [41], several data mining techniques are proposed to allow managers to make a better decision about salespeople performance measurement based on metrics defined by the business. The authors applied a naive Bayes model to classify salespeople into pre-defined categories provided by the business. They showed that the proposed approach can be applied in many companies using different KPIs.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tarasiuk, P.; Tomczyk, A.; Stasiak, B. Automatic Identification of Local Features Representing Image Content with the Use of Convolutional Neural Networks. *Appl. Sci.* **2020**, *10*, 5186. [CrossRef]
2. Xiao, H.; Qu, Z.; Lv, M.; Jiang, Y.; Wang, C.; Qin, R. Fast Self-Adaptive Digital Camouflage Design Method Based on Deep Learning. *Appl. Sci.* **2020**, *10*, 5284. [CrossRef]
3. Hung, S.C.; Wu, H.C.; Tseng, M.H. Remote Sensing Scene Classification and Explanation Using RSSCNet and LIME. *Appl. Sci.* **2020**, *10*, 6151. [CrossRef]
4. Neuhausen, M.; Pawlowski, D.; König, M. Comparing Classical and Modern Machine Learning Techniques for Monitoring Pedestrian Workers in Top-View Construction Site Video Sequences. *Appl. Sci.* **2020**, *10*, 8466. [CrossRef]
5. Neuhausen, M.; Herbers, P.; König, M. Using Synthetic Data to Improve and Evaluate the Tracking Performance of Construction Workers on Site. *Appl. Sci.* **2020**, *10*, 4948. [CrossRef]
6. Villaseñor, C.; Gallegos, A.A.; Gomez-Avila, J.; López-González, G.; Rios, J.D.; Arana-Daniel, N. Environment Classification for Unmanned Aerial Vehicle Using Convolutional Neural Networks. *Appl. Sci.* **2020**, *10*, 4991. [CrossRef]
7. Zhang, Y.; Yun, Y.; Dai, H.; Cui, J.; Shang, X. Graphs Regularized Robust Matrix Factorization and Its Application on Student Grade Prediction. *Appl. Sci.* **2020**, *10*, 1755. [CrossRef]
8. Gomede, E.; Miranda de Barros, R.; de Souza Mendes, L. Use of Deep Multi-Target Prediction to Identify Learning Styles. *Appl. Sci.* **2020**, *10*, 1756. [CrossRef]
9. Tsiakmaki, M.; Kostopoulos, G.; Kotsiantis, S.; Ragos, O. Transfer Learning from Deep Neural Networks for Predicting Student Performance. *Appl. Sci.* **2020**, *10*, 2145. [CrossRef]
10. Woo, H.; Kim, J.; Lee, W. Analysis of Cross-Referencing Artificial Intelligence Topics Based on Sentence Modeling. *Appl. Sci.* **2020**, *10*, 3681. [CrossRef]
11. Tatar, A.E.; Düştegör, D. Prediction of Academic Performance at Undergraduate Graduation: Course Grades or Grade Point Average? *Appl. Sci.* **2020**, *10*, 4967. [CrossRef]
12. Karlos, S.; Kostopoulos, G.; Kotsiantis, S. Predicting and Interpreting Students' Grades in Distance Higher Education through a Semi-Regression Method. *Appl. Sci.* **2020**, *10*, 8413. [CrossRef]
13. Li, Z.; Zhang, Q.; Wang, Y.; Wang, S. Social Media Rumor Refuter Feature Analysis and Crowd Identification Based on XGBoost and NLP. *Appl. Sci.* **2020**, *10*, 4711. [CrossRef]
14. Matosas-López, L.; Romero-Ania, A. The Efficiency of Social Network Services Management in Organizations. An In-Depth Analysis Applying Machine Learning Algorithms and Multiple Linear Regressions. *Appl. Sci.* **2020**, *10*, 5167. [CrossRef]
15. Chen, W.; Xu, Z.; Zheng, X.; Yu, Q.; Luo, Y. Research on Sentiment Classification of Online Travel Review Text. *Appl. Sci.* **2020**, *10*, 5275. [CrossRef]
16. Ali, M.; Baqir, A.; Psaila, G.; Malik, S. Towards the Discovery of Influencers to Follow in Micro-Blogs (Twitter) by Detecting Topics in Posted Messages (Tweets). *Appl. Sci.* **2020**, *10*, 5715. [CrossRef]
17. Choi, J.; Kim, Y. Time-Aware Learning Framework for Over-The-Top Consumer Classification Based on Machine- and Deep-Learning Capabilities. *Appl. Sci.* **2020**, *10*, 8476. [CrossRef]
18. Shafqat, W.; Byun, Y.C.; Park, N. Effectiveness of Machine Learning Approaches Towards Credibility Assessment of Crowdfunding Projects for Reliable Recommendations. *Appl. Sci.* **2020**, *10*, 9062. [CrossRef]
19. Gu, Y.; Shibukawa, T.; Kondo, Y.; Nagao, S.; Kamijo, S. Prediction of Stock Performance Using Deep Neural Networks. *Appl. Sci.* **2020**, *10*, 8142. [CrossRef]
20. Seo, M.; Kim, G. Hybrid Forecasting Models Based on the Neural Networks for the Volatility of Bitcoin. *Appl. Sci.* **2020**, *10*, 4768. [CrossRef]
21. Duarte, D.; Walshaw, C.; Ramesh, N. A Comparison of Time-Series Predictions for Healthcare Emergency Department Indicators and the Impact of COVID-19. *Appl. Sci.* **2021**, *11*, 3561. [CrossRef]
22. Liu, Z.; Chen, H.; Sun, X.; Chen, H. Data-Driven Real-Time Online Taxi-Hailing Demand Forecasting Based on Machine Learning Method. *Appl. Sci.* **2020**, *10*, 6681. [CrossRef]
23. Park, M.; Jung, D.; Lee, S.; Park, S. Heatwave Damage Prediction Using Random Forest Model in Korea. *Appl. Sci.* **2020**, *10*, 8237. [CrossRef]

24. Czibula, G.; Mihai, A.; Mihuleţ, E. NowDeepN: An Ensemble of Deep Learning Models for Weather Nowcasting Based on Radar Products' Values Prediction. *Appl. Sci.* **2021**, *11*, 125. [CrossRef]
25. Blachnik, M.; Kordos, M. Comparison of Instance Selection and Construction Methods with Various Classifiers. *Appl. Sci.* **2020**, *10*, 3933. [CrossRef]
26. Ryu, M.; Lee, K. Selection of Support Vector Candidates Using Relative Support Distance for Sustainability in Large-Scale Support Vector Machines. *Appl. Sci.* **2020**, *10*, 6979. [CrossRef]
27. Guo, Y.; Cao, X.; Liu, B.; Gao, M. Solving Partial Differential Equations Using Deep Learning and Physical Constraints. *Appl. Sci.* **2020**, *10*, 5917. [CrossRef]
28. Wieczorek, W.; Jastrzab, T.; Unold, O. Answer Set Programming for Regular Inference. *Appl. Sci.* **2020**, *10*, 7700. [CrossRef]
29. Wieczorek, W.; Unold, O.; Strąk, Ł. Parsing Expression Grammars and Their Induction Algorithm. *Appl. Sci.* **2020**, *10*, 8747. [CrossRef]
30. Kozłowski, N.; Unold, O. Anticipatory Classifier System with Average Reward Criterion in Discretized Multi-Step Environments. *Appl. Sci.* **2021**, *11*, 1098. [CrossRef]
31. ElMoaqet, H.; Kim, J.; Tilbury, D.; Ramachandran, S.K.; Ryalat, M.; Chu, C.H. Gaussian Mixture Models for Detecting Sleep Apnea Events Using Single Oronasal Airflow Record. *Appl. Sci.* **2020**, *10*, 7889. [CrossRef]
32. Kordos, M.; Boryczko, J.; Blachnik, M.; Golak, S. Optimization of Warehouse Operations with Genetic Algorithms. *Appl. Sci.* **2020**, *10*, 4817. [CrossRef]
33. Nusrat, A.; Gabriel, H.F.; Haider, S.; Ahmad, S.; Shahid, M.; Ahmed Jamal, S. Application of Machine Learning Techniques to Delineate Homogeneous Climate Zones in River Basins of Pakistan for Hydro-Climatic Change Impact Studies. *Appl. Sci.* **2020**, *10*, 6878. [CrossRef]
34. Kim, Y.H.; Moon, S.H.; Yoon, Y. Detection of Precipitation and Fog Using Machine Learning on Backscatter Data from Lidar Ceilometer. *Appl. Sci.* **2020**, *10*, 6452. [CrossRef]
35. Huynh, A.T.; Nguyen, Q.D.; Xuan, Q.L.; Magee, B.; Chung, T.; Tran, K.T.; Nguyen, K.T. A Machine Learning-Assisted Numerical Predictor for Compressive Strength of Geopolymer Concrete Based on Experimental Data and Sensitivity Analysis. *Appl. Sci.* **2020**, *10*, 7726. [CrossRef]
36. Wilmes, L.; Olympio, R.; de Payrebrune, K.M.; Schatz, M. Structural Vibration Tests: Use of Artificial Neural Networks for Live Prediction of Structural Stress. *Appl. Sci.* **2020**, *10*, 8542. [CrossRef]
37. Wang, L.; Li, X.; Wang, R.; Xin, Y.; Gao, M.; Chen, Y. PreNNsem: A Heterogeneous Ensemble Learning Framework for Vulnerability Detection in Software. *Appl. Sci.* **2020**, *10*, 7954. [CrossRef]
38. Pogiatzis, A.; Samakovitis, G. Using BiLSTM Networks for Context-Aware Deep Sensitivity Labelling on Conversational Data. *Appl. Sci.* **2020**, *10*, 8924. [CrossRef]
39. Kim, J.; Lee, K.; Choi, S. Machine Learning-Based Code Auto-Completion Implementation for Firmware Developers. *Appl. Sci.* **2020**, *10*, 8520. [CrossRef]
40. Nam, Y.W.; Cho, H.Y.; Kim, D.Y.; Moon, S.H.; Kim, Y.H. An Improvement on Estimated Drifter Tracking through Machine Learning and Evolutionary Search. *Appl. Sci.* **2020**, *10*, 8123. [CrossRef]
41. Calixto, N.; Ferreira, J. Salespeople Performance Evaluation with Predictive Analytics in B2B. *Appl. Sci.* **2020**, *10*, 4036. [CrossRef]

applied sciences

MDPI

Article

Automatic Identification of Local Features Representing Image Content with the Use of Convolutional Neural Networks

Paweł Tarasiuk, Arkadiusz Tomczyk * and Bartłomiej Stasiak

Institute of Information Technology, Lodz University of Technology, ul. Wolczanska 215, 90-924 Lodz, Poland; pawel.tarasiuk@p.lodz.pl (P.T.); bartlomiej.stasiak@p.lodz.pl (B.S.)

* Correspondence: arkadiusz.tomczyk@p.lodz.pl; Tel.: +48-42-631-39-57

Received: 30 June 2020; Accepted: 23 July 2020; Published: 28 July 2020

Abstract: Image analysis has many practical applications and proper representation of image content is its crucial element. In this work, a novel type of representation is proposed where an image is reduced to a set of highly sparse matrices. Equivalently, it can be viewed as a set of local features of different types, as precise coordinates of detected keypoints are given. Additionally, every keypoint has a value expressing feature intensity at a given location. These features are extracted from a dedicated convolutional neural network autoencoder. This kind of representation has many advantages. First of all, local features are not manually designed but are automatically trained for a given class of images. Second, as they are trained in a network that restores its input on the output, they may be expected to minimize information loss. Consequently, they can be used to solve similar tasks replacing original images; such an ability was illustrated with image classification task. Third, the generated features, although automatically synthesized, are relatively easy to interpret. Taking a decoder part of our network, one can easily generate a visual building block connected with a specific feature. As the proposed method is entirely new, a detailed analysis of its properties for a relatively simple data set was conducted and is described in this work. Moreover, to present the quality of trained features, it is compared with results of convolutional neural networks having a similar working principle (sparse coding).

Keywords: image representation; local features; autoencoder; convolutional neural network; machine learning

1. Introduction

Images are typically represented using regular grids of pixels. The information about image content is kept both in pixels' attributes (color channels) and, which seems to be even more important, in their spatial distribution. This kind of representation, although natural for humans, has at least one crucial drawback: It significantly complicates the design of effective computer algorithms able to accomplish tasks which are relatively easy for our visual system. The nature of this problem lies not only in the huge number of image elements and the variety of their possible distributions, but also in the fact that humans do not consciously operate on individual pixels. In most of the cases, the latter reason makes it impossible to directly write computer programs imitating the unconscious process of image understanding.

There are two typical approaches allowing to overcome the above problem. The first group of methods aims at changing and simplifying the representation of image content. The second one, instead of direct implementation, engages machine learning for this purpose. Although these methods can be used separately (simplified representations may allow to design algorithms ready for direct implementation and there are trainable models that can operate on raw pixels), they are usually combined together. The need for this combination originates in the limitations of existing models (e.g., specific input format) as well as in the necessity of selection of optimal model parameters (even carefully designed algorithms require problem specific fine-tuning).

An algorithm solving a given problem, created both with the use of machine learning techniques and without them, requires domain knowledge either to encode it in a computer program or to design the training objective. Unfortunately, domain experts usually have problems with sharing their knowledge in a form of ready-to-use mathematical formulas. They prefer to express it in imprecise natural language (it must be somehow adapted to become applicable in the source code of the program) or they provide it in a form of a training set (for a given input they specify the expected output). In both cases, it may be easier for them to operate on simplified representations rather than on millions of pixels. Training set preparation may be especially troublesome for some specific tasks. In particular, when precise image segmentation is the goal of analysis, the knowledge acquisition at pixel level can be tiresome and time-consuming. Moreover, if it concerns, e.g., medical applications, where the number of experts is limited, acquiring a huge and representative set of examples, required by most of machine learning algorithms, becomes almost impossible.

Another important aspect of image representation choice is interpretability of the algorithm results. Nowadays many artificial intelligence techniques have become a part of our life. Some of them are, or in the near future, will become responsible for our health and even life. Consequently, their authors must be prepared to at least explain their general principles to potential consumers to convince them that their product is safe. Operating on individual pixels makes it practically impossible. If instead the applied representation is significantly reduced and its components can be assigned a meaningful interpretation, such an explanation becomes plausible.

To conclude, the search for alternative image representations constitutes an important task from image analysis point of view. Moreover, it is very interesting itself as it may also allow better understanding of the principles of human visual system operation. There are many evidences that this system also tries to organize the recognition process creating several intermediate representations corresponding to elements of the observed scene [1]. Such intermediate representations are also observed in trained convolutional neural networks (CNN) [2–6] as they try to imitate the activity of visual cortex (to some extent). This was the main reason behind the choice of a CNN to automate the process of image representation construction in our research.

In this paper, we propose a new CNN-based method allowing to generate general image content representation. The image is described as a tuple of feature maps that describe the localization and intensity of selected visual features on the image plane. A feature map is a matrix that describes the intensity of a certain visual feature at every point of the image. This means that instead of global image features, such as *image is mostly blue* or *there are many vertical lines*, we focus on local features, such as *there is a vertical segment centered at a point with given coordinates*. Due to the local nature of the selected features, the proposed method encodes each feature occurrence as a single matrix element. The exact coordinates of the selected pixel reflect the precise localization of the feature in the image plane. As a result, the relation between feature maps and actual image content is much more direct than in the case of classic CNNs. This greatly simplifies the semantic analysis of features and feature maps, which originally required a specific approach such as deconvolutional neural networks [3].

Naturally, the aforementioned features have to be non-trivial. If visual features consisting of a single pixel were allowed, the trained feature extractor could take the form of an identity function and the feature maps would correspond to the original image. In order to select semantically meaningful features, our method ensures that the feature maps at a selected level are sparse matrices, where the neighborhood of each activated (non-zero) element is zeroed. The resulting image representation, based on the visual features of the image, should contain sufficient information to ensure that the goals of the analysis are met. The encoding is obviously lossy, but the most common patterns found in the training set are expected to be preserved. Because of its ubiquity in the field of machine learning, the MNIST data set of handwritten digits [7] was employed for the purpose of the present study. Not only does it allow for comparisons of the obtained results with those produced by similar techniques, but it is also, due to its simplicity, easily interpretable. The latter asset enables understanding of the meaning of the generated local features.

The remaining part of this paper is organized as follows. Section 2 discusses related works. In Section 3, the concept of convolutional neural networks is outlined and our novel contributions are defined. Section 4 describes the experimental framework for providing as sparse an image representation as possible, without losing the key information. The results are discussed in Section 5. A detailed analysis of the neural network models is illustrated with various visualizations. Finally, Section 6 presents the conclusions and directions for further research.

2. Related Works

The method of local feature identification proposed in this work uses the specific properties of CNNs. To enforce sparse coding, which allows us to determine the precise localization of these features, we propose a specific neural network architecture with additional filtering layers and a unique adjustment of the training objective. This is a novel approach, and thus it is hard to compare it with existing works. Nevertheless, in this section we try to present some of the related works aggregating them into three groups: works devoted to other methods of generating alternative representations of image content; works trying to automatically find semantic interpretation of features emerging in CNNs; and works having, to some extent, similar working principles to our approach.

2.1. Image Representation

As it was mentioned in Section 1, the change of image representation (extraction of features) is a crucial step in image analysis. It depends naturally on the type of considered task and consequently on the techniques that will be used. In the case of image classification tasks, global representations may be sufficient. However, for pattern localization, object detection and, in particular, for image segmentation, local features extraction is essential. Global representations treat the image as a whole and try to generate descriptors (usually feature vectors) which summarize colors, textures, shapes, etc. visible in this image. Features presented in this work are local. It means that the descriptors are assigned not to the whole image, but to specific locations (keypoints) within it. Naturally, raw pixels are also such local descriptors, but what we look for are reduced representations where the number of descriptors is significantly smaller than the number of all pixels. The reduced representation does not necessarily mean the loss of information. Their number is smaller but as they describe properties of image regions they may contain more information than color channel values assigned to single pixels. Moreover, additional information may be also kept in the data structure reflecting relations (including spatial relations) between these descriptors.

In the literature, there are two typical strategies for local descriptor finding: The first one uses segmentation techniques to define homogeneous regions of the image. Having found them, descriptors may be assigned either to these regions [8] or to their borders [9]. The regions are associated with information about their precise location (e.g., centroid of the region) and, consequently, their spatial relations can be discovered as well. The second strategy achieves a similar goal in the opposite way. First, characteristic points are sought for in the image plane (keypoints) and the local region around them is identified afterwards. In this group, such techniques as SIFT [10] or SURF [11] can be mentioned. They are particularly interesting, as they provide scale and orientation invariance. In all the above cases, after region or keypoint detection, descriptors (local features) must be computed. These descriptors can take into account the shape and the color of a region or they can be based on local gradients (SIFT) or wavelet responses (SURF). All of them, however, are designed manually by the author of a specific application.

The local features can be used both to classify image content and to solve more complex tasks. In the simplest case, clustered descriptors allow the identification of visual vocabulary depending on which bag-of-visual-words (BoVW) technique can be applied. In this approach, image content is transformed into a real vector (one-hot or frequency encoding) and consequently most classic pattern recognition techniques can be employed. If spatial relations between local features need to be taken into account or more complex tasks (object localization, segmentation) are to be solved, other methods must be used. For SIFT and SURF descriptors, a dedicated efficient matching algorithm was designed to find a correspondence of local features extracted from different images [10]. Other possible approaches construct a graph describing the image content, where local features are related to its nodes and spatial relationships are reflected in the edges. In such a case, geometric deep learning (GDL), allowing to generalize the CNN concept to non-Euclidean domains, can be applied [12–14]. Alternatively, active partitions [15], an extension of classic active contours, can be of use here as well.

2.2. Semantic Interpretation

There are many evidences that feature maps generated by successive convolutional layers of a CNN correspond with some semantically important parts of the analyzed images [3]. The identification of relationships between these parts and feature maps is not, however, a simple task. First of all, CNNs were always treated as trainable black boxes (similarly to other neural networks) and while designing their architecture no attention was paid to how the intermediate outputs can be interpreted. The resulting feature maps are usually blurred and it is really hard to understand the relation between them and the content of the input images. Moreover, in classic CNN architectures (pooling layers and no padding) the size of the feature maps is reduced in consecutive layers. This leads to further problems with identifying the precise location of semantically important regions.

In the literature there can be found several techniques trying to reveal the aforementioned relationships. In [4], first the peaks of the feature maps are mapped onto visual (receptive) fields within the input image. There their correspondence with known semantic parts is checked. As more than one feature map may be connected with a given part, a genetic algorithm is then applied to find the most appropriate subset of the feature maps from all the convolutional layers. In [5], instead of the layer outputs, their gradients maps, calculated with backpropagation algorithm, are used to find activation centers. In [16], the authors introduce class activation mapping (CAM), which can be used for identification and visualization of discriminative image regions, as well as for weakly supervised object localization. In that approach, a CNN network must be trained to classify images (supervised learning) and typical fully-connected layers are replaced by global average pooling (GAP) followed by a fully connected soft-max layer. The CAM for a given class can be found as a linear combination of the final feature maps generated by convolutional layers with weights corresponding to a specific network output. As the size of the class activation map is equal to the size of the final feature maps, it must be upsampled to be comparable with the input image. Finally, in [6], the authors assume that the top layers of a network correspond to the bigger parts or whole objects, while the lower layers reflect smaller parts which are building blocks for the more complex ones. They propose a method that is able to automatically discover a graph describing these relationships. It should be noted, that all these methods, although interesting, are quite complex. Moreover, they try to find correspondence with known parts (supervised process), which need not be optimal in every application.

2.3. Working Principles

The key part of the proposed method involves using sparse matrices as an intermediate step of image processing with CNNs. This should not be confused with sparse convolutional neural networks proposed in [17], as that work was founded upon using sparse filter matrices in multiple convolutional layers, whereas our approach is based on sparse outputs. Another method that applied sparse coding to CNNs was presented in [18] and addressed the problem of image super-resolution. This approach, however, also differs from ours in terms of both the main goal and the motivation behind using sparse matrices. In the present study, sparse matrices are utilized to generate image descriptions based on visual features.

These examples show that it is hard to find works with objectives similar to ours. Nevertheless, we were able to identify two groups of research areas which can be considered related and which will be used as a comparison base for our results.

Sparse coding in feed-forward neural networks was considered in multiple works as a tool for improving the performance of typical CNNs with dense matrices. This includes both theoretical analysis [19] and practical application to image reconstruction [20]. There are also multiple ways to generate sparse image representations for the tasks of image reconstruction and classification. One of the notable approaches is based on the Fisher discrimination criterion [21]. Multiple related works describe solutions based on CNNs [22–24], which make them similar and potentially comparable to the method presented in this paper. However, these studies are concerned with issues related to either the computational speed or accuracy and did not consider the problem of intermediate feature extraction. As presented in [25], sparse coding can simplify the classification task by maximizing the margin from the decision boundary in a selected metric space. It must be emphasized, however, that none of these works was dedicated to automatic detection of visual image features.

Sparse representation of the hidden layer outputs is also typical for spiking convolutional neural networks (SCNNs) [17]. The key component of SCNNs intends to simulate the electro-physiological process that occurs in synapses. Another notable advantage of SCNNs is the possibility to implement them on FPGA-based hardware [26]. The actual solutions are usually based on leaky integrate-and-fire (LIF) neurons [27,28] or spike-timing-dependent plasticity (STDP) learning [29]. Both above-mentioned methods involve the introduction of additional types of neurons that simulate spiking of the electrical charge, according to the selected model of synapses behavior. Distinct peaks related to the presence of specific patterns are rare enough to generate sparse data, which can be further reshaped into a sparse matrix. Thus, SCNN-based methods belong to the field of sparse coding. In the proposed approach, the learning process known from the basic CNNs is enhanced only with additional cost function components (which can be considered as model regularization) and activation functions, but no additional neuron types are introduced.

2.4. Contribution

The goal of our research was to create a tool that will be able to automatically (in an unsupervised way) discover spatially located visual features for a given class of images. These features should lead to a reduced representation of the image content without the loss of information contained there. Such a representation should allow to create image analysis algorithms which would be easier for interpretation, allowing the use of simpler models where external expert knowledge can be incorporated in a more natural way.

The image representation proposed in this work enables to achieve the above goal. Unlike the case of SIFT or SURF methods mentioned in Section 2.1, the feature identification does not rely on a manually designed algorithm, but it can be trained for a specific class of images. The role of a keypoint extractor is played by an encoder part of the proposed convolutional autoencoder. The value assigned to a given keypoint corresponds with the intensity of a visual feature. This value, together with the number of the sparse feature map where the keypoint was found, constitutes a form of a keypoint descriptor. It need not be more complex as no further matching is required when two images are compared. The feature map number directly identifies keypoints of the same type.

Although the types of the features (the numbers of the successive feature maps) seem to be very abstract, our approach allows us to discover and understand the nature of these features without the necessity of using such complex algorithms as those presented in Section 2.2. Their form can be revealed using a decoder part of our network. All of that would not be possible if we could not precisely locate these features in the original images, which is problematic for typical, blurred feature maps. Our approach solves this problem thanks to the novel training objective component, which enforces leptokurtic distribution of specific layer outputs, and thanks to the new filtering step added to the network architecture.

It should also be emphasized that the proposed representation differs significantly from reduced representations which can be obtained using classic feature reduction algorithms like PCA or non-convolutional autoencoder. The convolutional autoencoder used in this work takes the spatial relationships between reduced features and (which is its additional advantage) it performs feature reduction in a local way. The resulting features are calculated only on the basis of the pixels belonging to the respective receptive field. Such a weight sharing, typical for a CNN, reduces the number of trainable parameters and allows us to detect the same features in different places of the image plane regardless of the image size.

Finally, we use matrices, and not vectors, to encode images, because not only do we want to compress the information, but we want to extract the information about localization of the visual building blocks as well. This is a very specific application, which requires matrices only because the images are represented as regular grids of pixels. Nevertheless, the proposed approach is general. CNN is designed to work with matrix-like structures, but interpretation of these structures is irrelevant. If an autoencoder is used, one can obtain an encoder which generates sparse matrices preserving the whole information about the encoded data. In order to use the resulting sparse matrices, another processing tool unit must be designed. An example is a classifier described in Section 4.3. An alternative solution, which is not presented in this work, could be to train a CNN directly performing a specific task (e.g., classification) with enforced sparsity inside. In this case, however, the sparse information would not preserve the whole information about the input, but only this part which is required to accomplish a given goal.

3. Method

3.1. Method Overview

As proposed by LeCun [2], CNNs are feed-forward neural networks that typically consist of the following.

- Convolutional layers, which consist of multiple groups of matrix convolution filters. Input channels are convolved with corresponding filters from a group, and the sum of convolutions is a single output channel. The number of output channels is equal to the number of filter groups.
- Pooling layers, which divide the input image into a grid and reduce each cell to a single pixel. A commonly used pooling option is max-pooling, which takes the maximum value of each cell.
- Processing units such as activation functions or regularization techniques. The latter usually operates only on the gradient values used in the learning process, which can be used to implement additional components of the cost function.

In this paper, two applications of CNNs are considered. The most important architecture proposed in this study is a CNN-based autoencoder. Its primary goal is to minimize the difference between the input data and the output obtained for any input from the considered data set. The difference is calculated as the Euclidean distance between vectors of pixels. This implies that the resolutions of both input and output are expected to be the same. Thus, in our approach, no pooling layers are used and each convolutional layer is complemented by appropriate padding. As the convolution of $m_w \times m_h$ matrix with $f_w \times f_h$ filter yields $(m_w - f_w + 1) \times (m_h - f_h + 1)$ as a result, $(f_w - 1)/2$ zero-padding is added to the sides of the matrix, and $(f_h - 1)/2$ to the top and bottom. This is possible when both filter dimensions (f_w and f_h) are odd. This property is illustrated in Figure 1.

Without setting additional requirements, it would be easy to construct a perfect CNN-based image autoencoder. It would be sufficient that each layer generated an output equal to the input. This could be achieved with a convolution filter that has 1 in a single matrix element and 0 everywhere else. In order to avoid a meaningless result like that, we force one selected hidden layer to consist only of sparse matrices. The sparsity is guaranteed in the following way. For each non-zero element (i, j) of the output matrix, all other elements in $s \times s$ square centered in (i, j) are reduced to zeros. This step of data processing is further referred to as local-maximum filtering (LMF) (Figure 2) and its details are described in Section 3.3.

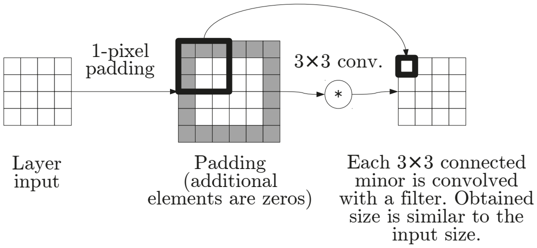

1-pixel padding
3×3 conv.
*
Layer input
Padding (additional elements are zeros)
Each 3×3 connected minor is convolved with a filter. Obtained size is similar to the input size.

Figure 1. The operation illustrated in this figure is a superposition of 1×1 zero-padding and matrix convolution with 3×3 filters. As the padding size matches the filter size properly, the output matrix has the same size as the input matrix.

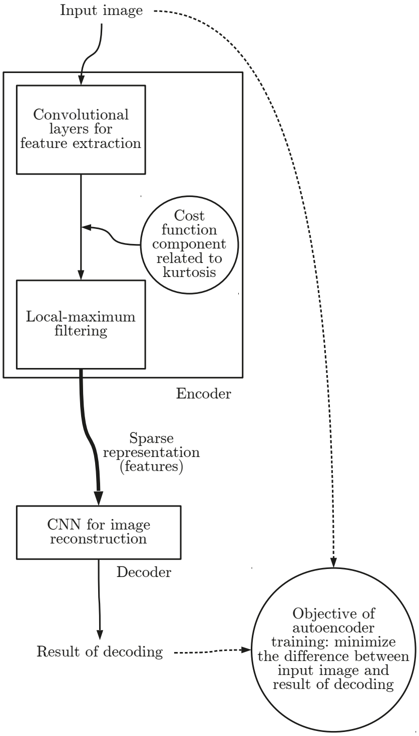

Input image
Convolutional layers for feature extraction
Cost function component related to kurtosis
Local-maximum filtering
Encoder
Sparse representation (features)
CNN for image reconstruction
Decoder
Result of decoding
Objective of autoencoder training: minimize the difference between input image and result of decoding

Figure 2. The full architecture of the autoencoder consists of two major parts: encoder and decoder. The encoder includes convolutional layers that can either be adjusted to the data set in the learning process or use some fixed weights. The result is further processed with local-maximum filtering (Section 3.3). In the case of adjustable convolutional layers in the encoder, the cost function related to the encoder's CNN output is modified in order to reduce the output kurtosis, as described in Section 3.2. The encoder output is fed into the decoder which consists of convolutional layers that participate in the learning process. The learning objective is to reproduce the original input image, while minimizing the reconstruction error, which is measured in terms of the Euclidean distance.

3.2. Leptokurtic Feature Maps

In order to obtain satisfactory results of local-maximum filtering in the selected hidden layerduring CNN training, the learning objective is enhanced with a kurtosis-based adjustment. It is based on the following observation; splitting the convolutional layer outputs into small subsets of highly activated points and low activation of the other elements may be equated to leptokurtic distribution of the outputs. Leptokurtic distribution (related to high kurtosis) means that all the elements are concentrated closer to the mean value than in the case of normal distribution. Forcing the leptokurtic distribution may be considered as a process equivalent to kurtosis maximization. The kurtosis function is continuous and differentiable almost everywhere, which provides the ability to apply gradient-based learning. Consequently, it can constitute an additional component of the cost function.

The kurtosis [30] of a vector $X = (X_1, X_2, \ldots, X_n)$ (this notation is valid for both random variables and fixed numbers) is defined as

$$\operatorname{Kurt} X = \frac{\mu_4(X)}{\sigma(X)^4} - 3, \tag{1}$$

where $\mu_4(X)$ is the fourth central moment and σ is standard derivation. In order to perform the gradient learning, we need to calculate the actual gradient. As the formulas are symmetric in terms of the elements of X, the only expression we need is

$$\begin{aligned} \frac{\partial(\operatorname{Kurt} X)}{\partial X_i} &= \frac{4\,(X_i^3 - \mathrm{E}((X - \mathrm{E}X)^3))\ \operatorname{Var}(X)}{n\ \operatorname{Var}(X)^3} + \\ &- \frac{4\ \mathrm{E}((X - \mathrm{E}X)^4) \cdot X_i}{n\ \operatorname{Var}(X)^3}. \end{aligned} \tag{2}$$

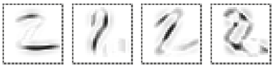
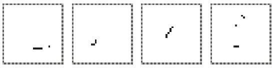
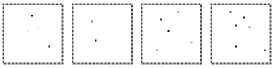
Formula (2) has one important disadvantage when used for gradient learning. As kurtosis (1) is indifferent to the magnitude of the inputs, the differential decreases as the magnitude of the inputs grows. As a result, big values would be modified more slowly by the learning process. In order to reverse this effect and obtain a change that is proportional to the current value of convolutional layer outputs (and to the corresponding weights—as in the case of CNNs these terms are proportional), the differential (2) is multiplied by $\operatorname{Var}(X)$. This means using exponent 2 instead of 3 in the denominators of expression (2).

3.3. Local-Maximum Filtering

The additional gradient component, which makes the selected part of the CNN yield leptokurtic outputs, does not guarantee the desired properties of the sparse output. In order to achieve literal sparsity, we need to make sure that some matrix elements are replaced with zeros. This could be easily achieved by thresholding—a process similar to that described in [31]. In order to limit the number of the remaining outputs, the threshold level could be defined as a quantile of the output of either the whole layer or a single resulting matrix. In this work, however, instead of using the global statistics of the CNN layer output, we propose a method that focuses on local properties.

The proposed approach, which generates only one non-zeroed element in each $s \times s$ matrix minor, has two major advantages. First, this operation is easy to implement for parallel computations, which is important, as the present CNN solution was implemented using GPU, supported by the Caffe framework [32]. Each element is considered separately, and is zeroed if it is not strictly the greatest element in the surrounding square. Another advantage of the proposed way of forcing the sparse representation is related to the interpretation of visual features. Typically, the input image has a continuous content, which means that the same visual feature is likely to be detected in multiple neighboring locations. Let us consider a horizontal edge visible in the image as an example. In the case of a long horizontal line in the image, the same local feature is obviously present in all the points of this line. If the sparse representation was related only to the number of activated pixels, there is a strong probability that we would obtain a subset of pixels forming a single connected component around the most visible feature (or, as in the example, along the line). LMF provides a direct solution to this problem. This situation is illustrated in Table 1.

Table 1. Local-maximum filtering (LMF) is a method that generates a sparse output, but is more practical than the standard thresholding. The points are designed to reflect the selected visual features of the image, and the local-maximum filtering makes it possible for each point to reflect a different occurrence of a feature. Thus, it is necessary to employ a mechanism preventing non-zeroed points from being located too close to each other. The strongest activated points are chosen in a greedy way, with only one point allowed in each $s \times s$ square. The presented illustration shows the result for $s = 3$.

The image	
Feature maps	
Threshold	
LMF	

3.4. Additional Thresholding

Local-maximum filtering, which was described in the previous section, generates an output that can be regarded as sparse. In the case of the MNIST data set [7], where input data consists of 28 × 28 images, local-maximum filtering with radius $s = 3$ ensures that at most 49 elements of each 28 × 28 matrix remain non-zeroed. This means that either for the original data set or any larger input images, at most 6.25% elements of the output data have values other than zeros. It may be expected, however, that many of the 49 elements have insignificant, near-zero values anyways. Local-maximum filtering yields such an element in each isolated region of the image plane, even if the corresponding visual feature is not present in that region.

It is difficult to suggest any general purpose threshold for the selection of the significant points, as it may depend on the weights of the convolutional layer and on the context of the considered image. In some of the experiments that involve the original MNIST data set, where each image presents exactly one object, we manually limit the number of points in each matrix that are used to encode that object. After local-maximum filtering, which prevents the points from being located too close to each other, all the points except the k highest values are replaced with zeros. For $k = 5$ and $k = 3$, it yields 0.64% and 0.38% non-zero values, respectively. This is equivalent to image thresholding, with the threshold value dependent on the appropriate quantile of the values from the processed matrix. This highly sparse representation can be used to experimentally determine how much information is actually preserved in the small number of points.

3.5. Properties

The sparse output obtained from the selected hidden CNN layer can be considered as a form of image encoding, as it is supposed to be used by subsequent layers to reconstruct the input image. Thus, a neural network architecture that meets the presented assumptions can be considered as a general purpose tool for sparse image encoding. The encoding is based on local visual features of the image, which may be easily explained as follows. One of the commonly known properties of

CNNs is the invariance to translation of the visual features of objects on the input image plane [2]. The translation of the object automatically results in a similar translation of its representation generated by the convolutional layer. This is essentially true for a single matrix convolution and remains relevant for sums and superpositions thereof. The outputs of the convolutional layers are known as feature maps, as CNNs take a biological inspiration from the visual cortex [2,33]. An important aspect is the size of the feature or object visible in an image under examination. Calculations performed in order to obtain each element of the feature map involve data from a specific range of the input image, known as the visual field. In the case of the initial convolutional layers, the visual field is significantly smaller than the image itself—for the first layer, it is simply equal to the filter size. If the visual fields of the layer with a sparse output contained the whole input image, whole objects could be encoded as single pixels. However, this would be equivalent to an image classification task, without the analysis of particular elements of the recognized object. In order to split the original image into more basic features, we use visual fields that are smaller than the image itself. In one of the examples, presented in the following section, we use 14 × 14 visual fields selected from 28 × 28 input images. The sizes of visual fields of a neural network are easy to estimate, particularly if the network consists of convolutional layers only, an example is shown in Figure 3.

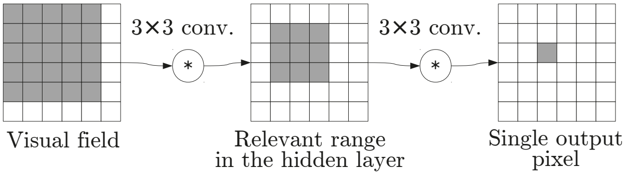


Figure 3. Each output pixel depends on multiple elements from the previous layers. The scope of the related pixel from the previous convolutional layer matches the size of the convolutional filter applied. By tracking the dependencies back to the input data matrix, we can determine the size of the visual (receptive) field. The convolutions involve one-pixel padding from each side, so the image size does not change. A single output pixel is calculated on the basis of 3 × 3 minor of the hidden layer, and the size of the visual field is 5 × 5.

4. Experiments

4.1. Feature Identification

The experiments described in this section were performed on the original MNIST data set [7], which offers the advantages of a large number of images, a resolution that makes the computational cost considerably low (28 × 28 pixels), and a simple semantic interpretation of the results, as the samples contain handwritten digits.

Three experiments were performed in three set-ups that implemented the idea presented in Figure 2. According to the original partition of the MNIST data set, the autoencoder models were trained with the 60,000 training samples, while the separate 10,000 samples were used for the evaluation. The models were different in terms of the visual features used to encode the image. The architecture of the decoder part, described in Table 2, was common for all the models. None of the models used any form of pooling, and the coexistence of filters and paddings made the matrix size remain unchanged throughout the layers. The presented models applied typical techniques associated with CNNs, such as the dropout method [34] and PReLU activation functions [35]. The encoders were designed as follows.

- `MF4`: Four manually designed features were used. The filters were fixed and no learning was performed on this encoder. The features were related to vertical, horizontal, and diagonal lines (in both diagonal directions). The contents of the proposed feature-detecting convolution filters are presented in Table 3. The filters applied in this experiment are of a very generic character, so no advantage may be drawn from using specific filters that fit the data set. Absolute values of the matrix convolution results are used, which is followed by local-maximum filtering with radius 3, as described in Section 3.3.

- AF4: Four automatically computed features. The architecture of the encoder (or feature extractor) is described in Table 4. The size of the visual field (Section 3.5) for this architecture is 14 × 14. The special activation function used in the last layer of this architecture is denoted as ENCODE. The outputs of Enc5 are tuples of four sparse matrices, considered as extracted features of the input image.
- AF5: Five automatically computed features. This experiment is largely similar to AF4, but five matrices are generated as the output of Enc5 layers (instead of four in AF4).

The encoder and decoder could be considered as separate utilities, but combining them into one neural network model made it possible to actually train the feature detectors in AF4 and AF5 experiments. The training was aimed at minimizing the total square error of the autoencoder.

The MF4 features are the most natural approach, as the features were designed manually in order to approximate any pattern that consists of thin lines. The four basic directions, shown in Table 3, fit the structure of a filter matrix precisely. Any change to this approach, such as a set of 3 or 5 segment-based features, would require an arbitrary choice of a direction and involve a specific approximation when described as convolutional filters.

As MF4—a solution with 4 kinds of features—was selected for its simplicity, the most direct comparison based on the automatic features identification involves 4 features as well, which is demonstrated by the AF4 set-up. However, automatic detection of features does not directly indicate any specific number of features as correct. The design of 5 equally important features for MNIST is unintuitive, but the potential gain can be easily researched for using the automatically trained encoder. The AF5 set-up was introduced for this purpose. The number of features can be expanded arbitrarily further, but as the number of features would grow, they would be increasingly difficult to visually distinguish. For the purpose of visual presentation of the results, we focus on a maximum number of 5 features. However, if the data set was more complex than MNIST or involved color images, it may be crucial to introduce more features.

The specifics of automatic encoder training process that are described in Table 4 were proposed as a compromise. This architecture is complex enough to identify potentially useful image features while avoiding the possible disadvantages of overly complex models, such as high resources usage and duplicated filters. The number of layers and the filter sizes were defined by the requirements on the visual fields, while the number of filters in each layer was selected by trial and error. The results were roughly convergent around the chosen preferred values. The possible changes obviously include permutations of feature detectors in the encoder output. The full training time was long enough to make the detailed parameter tuning remarkably difficult, but we believe that the presented models are sufficient to demonstrate the properties of the proposed methods.

The complete encoder architecture from Table 4 involves 28,570 adjustable parameters for AF4 and 29,570 for AF5. The slight difference is related to the last convolutional layer in the sequence. Remaining in a similar order of magnitude, the total number of decoder weights was 114,450 for AF4/MF4 and 115,200 for AF5, due to the additional filter group in the first convolutional layer. While the training process was relatively complex, we believe that the final model can be described as lightweight.

It must be emphasized that getting the optimal autoencoders available to this method would require much more detailed fine-tuning and repeated experiments. However, the presented demonstration of the method does not require putting this kind of endless effort to the optimization. We have defined three different set-ups, which are going to be useful for the analysis, and we use fixed training conditions for all of them, so we can adequately compare them with one another.

The autoencoder error was calculated as the difference between the input and the expected output. Data from the MNIST data set [7] could be considered as a set of 8-bit grayscale images with brightness levels varying from 0 to 255. However, the presented results refer to normalized values from the [0, 1] range. This applies to the average errors from Table 5. The error for a single sample is a half of the sum of quadratic errors for all the pixels. The table presents average errors for a certain set of samples—both for the whole data set and for all the digits considered separately. The autoencoder itself, in accordance with the previously described architecture, did not use any information on object classes while training.

Table 2. All the experiments (namely, `MF4`, `AF4`, and `AF5`) related to the general architecture of the autoencoder, which is shown in the Figure 2, use the same layout of the decoder part. This table includes a detailed layout of the convolutional layers and the activation functions used in the decoder, such as PReLU [35]. The rows of the presented table describe consequent layers of the decoder CNN, denoted as `Dec1`–`Dec4`.

#	Outs	Filter	Pad	Dropout	Activation
`Dec1`	30	5 × 5	2 × 2	-	PReLU
`Dec2`	30	5 × 5	2 × 2	-	PReLU
`Dec3`	30	5 × 5	2 × 2	0.5	PReLU
`Dec4`	30	5 × 5	2 × 2	0.5	PReLU

Table 3. One of the autoencoder-related experiments, labeled as `MF4`, uses fixed encoder filters (the encoder part of the overall architecture is described in Figure 2). This table presents the predefined values of 7 × 7 convolutional filters, visualized as bitmaps.

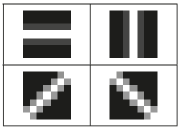


Table 4. Automatic feature extraction experiments, labeled as `AF4` and `AF5`, use adjustable encoders (Figure 2) with multiple convolutional layers. The layout of the layers and the corresponding activation functions (including PReLU [35]) are presented. The ENCODE activation function is a short term for a sequence of operations: the PReLU activation function, the absolute value, the layer that modifies gradients with relation to kurtosis (Section 3.2), and local-maximum filtering with radius $s = 3$.

#	Outs	Filter	Pad	Dropout	Activation
`Enc1`	50	3 × 3	1 × 1	-	PReLU
`Enc2`	30	3 × 3	1 × 1	-	PReLU
`Enc3`	30	3 × 3	1 × 1	-	PReLU
`Enc4`	30	3 × 3	1 × 1	-	PReLU
`Enc5`	4 or 5	5 × 5	2 × 2	-	ENCODE

Table 5. The autoencoder errors, measured in terms of the Euclidean loss function, were calculated for all the proposed network architectures. In addition to the general error on the test set from the MNIST data set [7], specific values were calculated for each class separately. Thus, it was possible to evaluate how well the selected visual features described each of the digits.

Class	`MF4`	`AF4`	`AF5`
0	14.624	9.852	8.820
1	7.248	5.056	4.500
2	12.210	9.308	9.386
3	11.726	8.348	7.966
4	10.262	8.136	7.760
5	12.168	8.876	8.114
6	11.802	9.372	8.780
7	9.986	6.592	6.266
8	12.656	10.458	10.198
9	10.260	7.682	6.732
All	11.220	8.304	7.792

The results presented in Table 5 prove a relative success of all the experiments. It is worth noting that the maximum quadratic error between 28 × 28 matrices is 784, and the expected quadratic difference between matrices of uniform random $[0, 1]$ elements is 130.67. The errors obtained from the experiments presented are lower by a whole order of magnitude (per-subset average errors are more than 11 times smaller than the mentioned estimation). The only limitation, which leads to the presumption that error of zero is impossible for the MNIST data set, is based on the sparse encoding that needs to be used as an intermediate sample representation. Due to the specific properties of this sparse representation, there is no other comparison. The training of the decoders was performed in a unified way for all the set-ups, so the results from Table 5 reflect the usefulness of features selected by the encoders. Therefore, in absence of more general ground truth, `MF4` results can be considered as reference values for evaluation of `AF4`- and `AF5`-based features.

The first conclusion is that the features specific to the data set performed better than the generic manual suggestion—the `MF4` experiment resulted in the highest autoencoder errors. The difference between the results of the automatic variants with 4 and 5 features appears to be slight when compared to `MF4`. It may be also concluded that using a higher number of features makes the encoding more precise, i.e., it enables preserving more information about the exact contents of the original image. Surprisingly, the results for digit 2 are slightly better in the case of `AF4` than in `AF5`, which is an exception to the mentioned rule.

The differences between classes can be explained by the geometric properties of the digits. Digit 0, which is round, generated a particularly high error in `MF4`, as lines of fixed directions made it difficult to recognize round shapes. The error for 0 in `MF4` was even higher than for 8, which contains crossing diagonal line segments in the center—the direction of these segments apparently fits the designed filters. Remarkably, the lowest errors for `MF4` were obtained for digits that literally consist of straight segments, namely, 1 and 7. While digit 9 was the third best, 4 was the close fourth, which fits the pattern, as 4 consists of long segments and 9 has a small circular head and a straight, long tail.

The comparison of `AF4` and `AF5` error rates provided a number of other important observations. In both experiments, 8 was the worst case, which can be justified by the most visually complex shape—a single line that crosses itself and forms two circles is especially difficult to describe with features obtained as the result of convolutional filters. The other digits with significantly high error rates were 0, 2, and 6. For `MF4`, the digits that contained circles (0 and 6) produced high error rates, while for `MF5`, the second worst case was 2. It suggests that `MF5` was able to handle the features characterized with small circle shapes better than MF4, partly at the cost of segments specific to digit 2. The difference between errors obtained in MF4 and MF5 is the highest for 0, 9, 5, and 6; remarkably, three of these digits have shapes containing circles.

As was the case in MF4, and also in the other experiments, the lowest error rates were generated for 1 and 7. The property of these digits, which can be summarized as *having a simple shape*, seems to be pretty universal, as confirmed by the results obtained for AF4 and AF5.

4.2. Feature Reduction

The experiments presented in the previous section involve local-maximum filtering, which ensures that at most 6.25% of matrix elements are non-zeros. In this section, however, the results related to even higher levels of sparsity are considered. The number of zeroed elements in the encoding is increased, but exactly the same decoders, trained in Section 4.1, are used to generate the results presented below.

Figures 4–6 include the results of sparse matrices decoding for MF4, AF4 and AF5 experiments respectively. Each table includes the following.

- The original encoding errors (for comparison).
- The result achieved with each matrix being greedily reduced to 5 highest values, and all the other elements being replaced by zeros. The description of there results consists of an absolute encoding error and a relative increase (compared to the first column).
- The results of an experiment similar to the previous point, but with 3 points instead of 5.

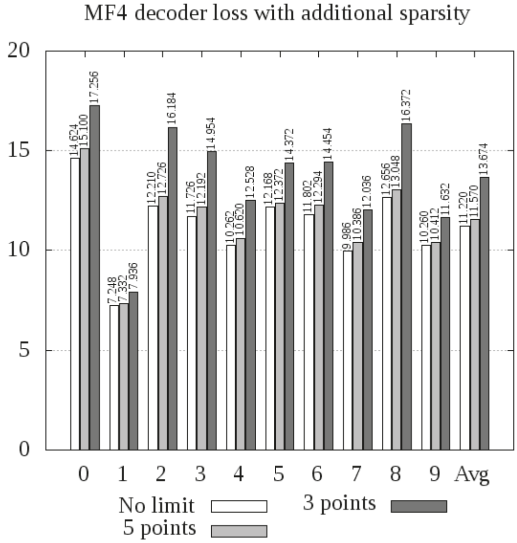


Figure 4. The pretrained decoder from the MF4 model can be used either with the original data without a specific limit of non-zero elements in the encoding, or with modified encodings, where each matrix contains up to 3 or 5 non-zero elements. The plot presents decoding errors for images showing individual digits and for the whole test set.

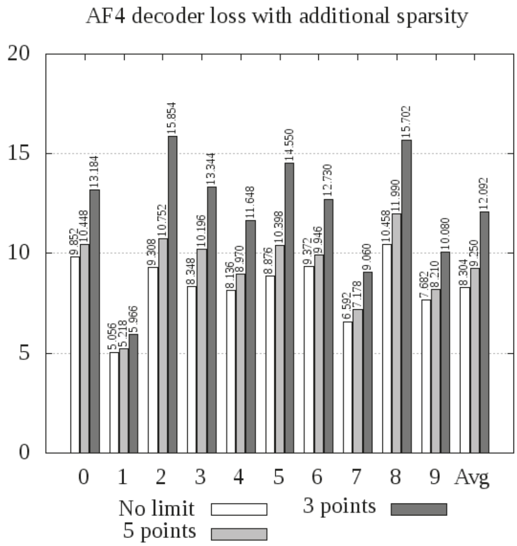


Figure 5. The pretrained decoder from `AF4` used for decoding of both the original and the highly sparse data, as in Figure 4.

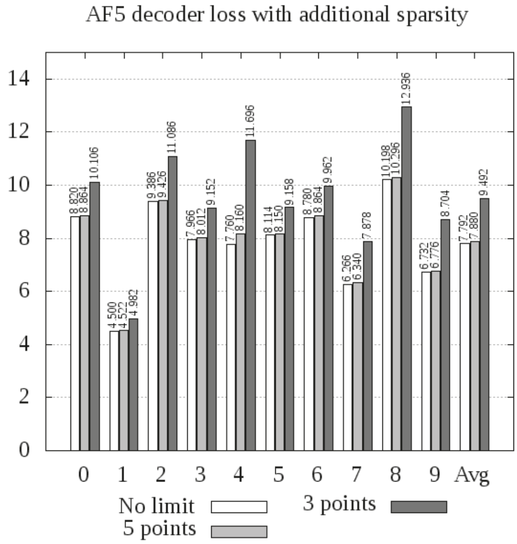


Figure 6. The pretrained decoder from `AF5` used for decoding of both the original and the highly sparse data, as in Figure 4.

As we can conclude from Figures 4–6, experiment `AF4` seems to be most sensitive to additional thresholding, which is particularly evident in the case of encoding digits 2, 3, and 5. However, the other experiments, namely, `MF4` and `AF5`, behave in quite a similar way, giving slightly above 20% greater average loss when 3 points per matrix are used, and only a few percent in the case of 5 points.

The most remarkable phenomenon related to experiment AF5 is the sensitivity of digit 4 to sparse autoencoding. The autoencoder error increased by 5% for 5 points and above 50% for 3 points. This leads to the conclusion that digit 4 consists of a greater number of visual feature occurrences than any other digit, and omitting some of these features generates a significant error.

The most important conclusion is that further sparsity enforcement is generally acceptable, unless the features are too specific (AF4 case) and the reduction is too great (3 points case). With 5 points per matrix, both MF4 (error increase: up to 4%, 3% in average) and AF5 (error increase: up to 5%, only 1% in average) models yielded acceptable results. This means that the whole 28 × 28 digit can be compressed into 20 points (in the case of MF4) or 25 points (AF5), with encoding errors presented in Figures 4 and 6.

4.3. Classification

In order to determine how much information was preserved in the encoding, we attempted to decode the original image, as described in the previous sections. However, it is not the only possible approach. It is debatable whether the Euclidean distance between the autoencoder output and the original image may serve as a reliable tool for measuring the loss of significant information in the encoding process. However, regardless of the Euclidean distance value, the encoding can be considered as useful if it is sufficient to determine the originally encoded digit. This property can be tested in the image classification task using pre-generated encodings. Another reason for performing this experiment is the possibility to discuss the relation of our results to the numerous classification results from the literature, where a similar task was performed on the same data set.

The sparse representations obtained from the encoder (according to the description shown in Figure 2) can not only be used to decode the original digit, but also directly in the image classification task. All the experiments (MF4, AF4, and AF5) were performed on the basis of a CNN classifier architecture proposed by the authors of this study. The classifier consisted of 6 convolutional layers and two hidden fully connected layers. The last convolutional layer and the hidden fully connected ones were trained using the dropout method [34]. Such an approach was decided, as it should provide adequately complex classifier model to achieve fine results without defining a very deep neural network which would require specific approach to the problem of a vanishing gradient. A model with 30 convolutional filters in each layer and 500 neurons in the hidden fully connected layer was selected as a point where no further extension improved the result significantly. The presented values indicate that the trained classifier models consisted of less than 50,000 convolutional parameters and approximately 12 millions of weights of the fully-connected layers. It must be emphasized that finding the optimal classifier model was not the key objective of this paper. The selected classifier configuration is possibly similar for all the encodings, and the results are well adjusted to the task of comparison between the setups. Further effort to optimize such a classifier remains possible, but this issue alone definitely exceeds the scope of this paper.

For the classification tests, the data set was divided into a testing set (10,000 samples) and a training set (60,000 samples), as proposed in the original MNIST [7] database. Each classifier was tested with a representation obtained by a specific autoencoder. This architecture made it possible to perform additional experiments. Instead of a raw encoder output, where up to 49 pixels from each matrix could have positive values, manually thresholded matrices were used in order to eliminate near-zero values. The data prepared in this way are used in the same tasks as described in the previous sections. It must be emphasized that the same classifier models were used for both the original encodings and the thresholded versions. All the results are presented in Table 6.

Table 6. Encodings from Section 4.2 were tested in the image classification task. Each encoding was used both in the original form, obtained as a result of local-maximum filtering, and in the reduced form that guarantees additional sparsity (Section 3.4). For each encoding (MF4, AF4, and AF5) a separate classifier was trained. The results for the additionally thresholded data were generated with the same neural networks that were trained for the original encodings.

Model	Accuracy
MF4 (3 points)	93.84%
MF4 (5 points)	95.59%
MF4	95.64%
AF4 (3 points)	96.86%
AF4 (5 points)	98.33%
AF4	**98.67**%
AF5 (3 points)	**97.88**%
AF5 (5 points)	**98.40**%
AF5	98.40%

As we can deduce from Table 6, the accuracy of the classifier seems to reflect the autoencoder error from the previous tables. Thus, MF4 results are clearly the worst—the general features are not nearly as useful as the automatically calculated ones that were used in experiments AF4 and AF5. The only surprise is that the AF4 classifier on the full encoder results was the best from the whole table (98.67%)—the difference is slight, but the classifier related to AF5 made more mistakes. However, when the reduced representations are considered, the sensitivity of the representations encoded in AF4 to the additional thresholding is clearly visible, as was the case with the autoencoder. AF5 representations, when reduced to five points per matrix, resulted in as good results as in the case of the original classifier objective, providing an accuracy of 98.40%. Surprisingly, the representations reduced to 3 points per matrix, despite generating over 20% higher autoencoder error, can still be regarded as acceptable for practical applications, as even with so drastically reduced information the classifier is able to recognize the digit correctly in 97.88% of the cases.

As the results from Table 6 are denoted as classification accuracies that can be easily compared to each other, we can seek comparison with other MNIST classifiers from the literature as well. However, it must be emphasized that in this paper we treat the image classification just as an analytic tool, and not as the key objective of this paper. Presumably, using the raw MNIST images to train a classifier, without the added difficulty of sparse representation, could only improve the achieved accuracy. The general problem of MNIST classification can be solved with accuracy as great as 99.79% [36]. We do not pursue to beat this result. For broader perspective, we can discuss the relation of our results to the other state-of-art MNIST classifiers that somehow involve sparse representations. Due to the varying objectives and circumstances, such comparisons require analysis that exceeds the straightforward competition for the best accuracy.

The results from Table 6 are clearly better than the classification results obtained with the classical approach to sparse representations and dictionary learning. This includes particularly the convolutional sparse coding for classification (CSCC) method presented in [23], which achieved an accuracy of 84.5% on the MNIST data set, outperforming many previous approaches to sparse representations and dictionary learning. It must be emphasized, however, that the problem statement of that paper was not the same as ours. Dictionary-based methods are more computationally complex. Moreover, in [23], only the training subsets of 300 images were used. Thus, while that work may be regarded as an interesting reference for the present study, a direct comparison would be inappropriate.

Another remarkable work on sparse representation was based on the idea of maximizing the margin between classes in the sparse representation-based classification (SRC) task [25]. The sparse representations related to this model were strictly related to the classification task. In contrast to that approach, the method presented in this paper does not use any information on object labels when training the encoder. On the other hand, no convolutional neural networks were used in [25], and some solutions used in that paper might be outdated. The best classifier presented there reached a 98.13% accuracy rate. This result is lower than `AF5` with 5-point-based reduction, which is already very sparse.

The CNN-based architecture ensures that the image features are detected is a translation-invariant manner; translation of a feature would entail translated coding. A similar concept was applied in [22], which proposed another approach to CNN-related sparse coding. The results of MNIST classification were generated for both the unsupervised and the supervised approach to sparse coding, with 97.2% and 98.9% accuracy rates, respectively. It must be emphasized, however, that the method shown in the present paper should be considered as unsupervised, as the autoencoder does not use information on the object labels. The size of the input data is not fixed—the method works for any input data, irrespective of the number of rows and columns. Thus, we cannot speak of a class that an input belongs to and some valid input images can contain multiple digits, which makes it impossible to assign them to a single label.

The results of MNIST classification that are somehow related to the idea of using sparse coding in the hidden layers in image processing tasks are also known from the works on spiking neural networks. A solution which involved weight and threshold balancing [31] performed reasonably well, resulting in a 99.14% accuracy rate in a method that combined spiking neural networks and CNNs. However, the method proposed in [31] was very complicated and the image representations that it produced were not as sparse as those presented in this paper. Similar remarks hold with respect to the work in [28] (non-CNN spiking network with LIF neurons) and the work in [31] (bio-inspired spiking CNNs with sparse coding), which achieved the accuracy of 98.37% and 97.5%, respectively. The latter approach is particularly interesting, as it was coupled with a visual analysis of features recognized by the neural network. The accuracy rates achieved were slightly lower than these obtained in this paper. However, the results in [31] cannot be directly compared with these achieved in this study because of differences in the architectures proposed. Moreover, the work in [31] involved an additional learning objective—the classifier was designed and trained to handle noisy input.

4.4. Larger Images

All our previous experiments were related to the original MNIST data set [7], where each sample was a 28×28 image displaying a single centered digit. The autoencoder was designed to encode each digit in a way that enabled as accurate a reconstruction of that digit as possible. As the solution is based on CNNs (both encoder and decoder, as it is shown in Figure 2), the whole mechanism is translation invariant—-a translated digit would simply yield a translated sparse representation. What is more, as no pooling layers are used, the model can be successfully applied to images of any size. Both matrix convolutions and element-wise operations will still be possible to be computed.

The modified data set with larger images was prepared to illustrate this property, as shown in Figure 7. The digits were placed on 80×80 plane in a greedy way, as long as placing another non-intersecting 28×28 square was possible. The test set consisted of 2000 images: 68 with a single digit, 119 with two digits, 888 with three digits, and 925 with four digits each.

The features described in the proposed sparse representation are deliberately smaller than the whole digits, so our model should not be considered as digit classifier, in particular for larger, more complex images. Nevertheless, digits should be reconstructed equally well regardless of position and context. The experiment introduced in this section is intended to demonstrate this property.

Table 7 shows the average per-digit autoencoder errors for the extended data set. In the case of images with multiple digits, the error was divided by their count. The division into separate classes was impossible, as a single large image was likely to contain multiple digits from different classes. The overall conclusion is that the `MF4` model is quite sensitive to the behavior of image boundaries and, while useful, produces almost 10% greater errors in this atypical application. The models with automatically calculated features—AF4 and AF5—provided a very slight error increase when compared to the original task. This confirms the universal nature of the presented autoencoders. As expected, translational invariance makes it possible to describe the translated objects as easily as the original inputs. The application of extended input sizes does not create any technical difficulties either.

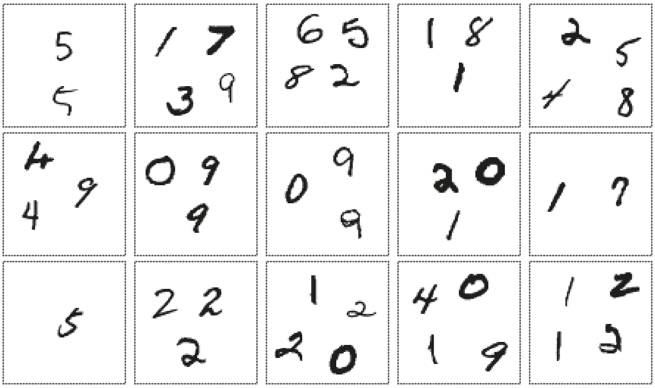


Figure 7. In order to demonstrate that the proposed autoencoder architecture is size-independent, 80 × 80 images containing multiple digits from the MNIST data set [7] were generated. The extended images contain up to four objects.

Table 7. The autoencoders trained in Section 4.1 were used with larger images, as shown in Figure 7. An average per-object error was calculated and compared to the original results, related to the objects with a single centered object. The information on the relative error increase is included in the table as well.

Model	Original	Large	Increase
MF4	11.220	12.220	8.91%
AF4	8.304	8.342	0.46%
AF5	7.792	7.902	1.41%

5. Analysis

The autoencoder architectures presented in Section 4.1 can be further analyzed in terms of errors and semantic understanding of related visual features. The results from Sections 4.1 and 4.2 can be used to compare the overall quality of selected solutions and analyze the dependency between the autoencoder errors and particular image classes. This could be related to the level of adjustment of the set of selected features to the data set, for example, manually selected directions of lines are particularly irrelevant in the case of digits 0 and 5. Another important aspect is the inner complexity of the digits. Digit 1, which usually consists of one or two segments, is especially easy to model. The exact directions of the segments, however, do not fit any of the manually selected filters. Thus, only an automatically computed feature extractor was able to take full advantage of the simplicity of the shape of this digit, producing a remarkably low error rate for this class. The errors for the other nine classes differ only slightly in the case of the automatically chosen features, which means that this solution actually reflects the properties of the data set. Manually selected features were not digit-specific, which resulted in generally higher error rates and greater variance of errors among different digits in MF4 experiment.

In the case of the manually designed features, the convolution filters were created arbitrarily, as illustrated in Table 3. However, another way of visual presentation of the features can be achieved by using the decoder part of the autoencoder architecture (see Figure 2). The results of decoding single points related to each of the manually chosen features are presented in Table 8—the relation to the filters shown in Table 3 is apparent. This technique of visualization can also be applied to the models with the automatically constructed features. The results of this approach are reported in Tables 9 and 10—apparently, this time the features are implicit and difficult to categorize semantically.

As the shape of each visual feature is complex, and possibly context-dependent, we need an analysis that goes beyond simple visualization. Tables 11–13 provide information on the intensity of each feature in the data set, including both averaged results and those obtained for separate subsets consisting of different digits. The intensity is calculated as a sum of the elements of the respective encoding matrix (output of the encoder, as illustrated in Figure 2). It must be emphasized that because of different weights in neural network models related to each decoder/classifier, it is possible to compare only the values within the same table. However, the results presented still enable a thorough analysis that otherwise would be difficult to perform.

The MF4 results (see Table 9) are especially easy to understand, as digits are rather taller than wide, vertical lines (feature #2) are the most visible among the data set. Remarkably, digit 1 contains almost no other features. The least intense feature is related to the backslash segments (#4), which occur mostly as a part of arc (digits 8, 3, and 0). As it was expected, digit 8 is especially rich in all kinds of features. However, because of the typically skewed writing style, even in this particular case backslash lines are less intense than segments in the other directions.

The features selected in the AF4 model are particularly interesting. It is relatively clear that no feature is dedicated solely to backslash lines, which is demonstrated in Table 10. Instead, we get feature #1 that seems to address the right side of a small arc. This is reflected in Table 11—digits 8, 6, and 3 exhibited particularly intense occurrences of this feature. While features #2 and #3 seem to reflect vertical and horizontal lines, respectively (digit 1 is strongly correlated with feature #2), the relation is not nearly as straightforward as it was in MF4. Feature #4, related to slash lines, seems to reflect a part of digit 2 especially well.

The results of the experiment AF5, which yielded the best autoencoder (Section 4.1) and classification accuracy for highly sparse data (Section 4.3), are less intuitive. The idea of horizontal lines is divided between features #2 and #4. Features #1 and #5 seem to handle both slash/backslash lines and sections of small arcs as well—both these features are important for encoding digit 8. Feature #3 clearly describes some cases of curves that are oriented vertically (digits 2, 8, and 6), but it does not involve straight segments. Digit 1 is described mostly with feature #4. Visual features detected by AF5 model are mostly implicit and difficult to describe semantically.

Table 8. MF4 decoder results for synthetic encoding, where only one point is activated. This is intended to show the shape of the visual features that are described by each matrix.

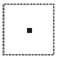
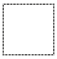
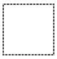
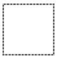
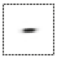
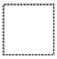
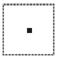
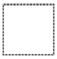
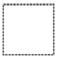
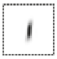
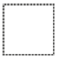
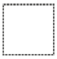
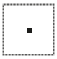
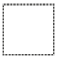
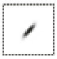
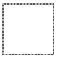
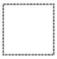
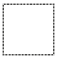
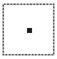
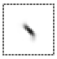
	Code				Output
#1					
#2					
#3					
#4					

Table 9. For each visual feature MF4, the sum of occurrence intensities (values in the encoding) was calculated. The result was considered separately for each digit, as different digits consisted of different visual features. It must be emphasized that the results in this table are relative and, while comparing them to each other is noteworthy, they should not be compared with the results from the other tables.

	#1	#2	#3	#4
0	1.59	1.82	1.88	1.86
1	0.52	1.53	0.88	0.87
2	2.16	1.69	1.89	1.79
3	2.28	1.69	1.94	1.90
4	1.41	2.29	1.70	1.50
5	2.30	1.58	1.82	1.78
6	1.67	1.84	1.78	1.55
7	1.42	1.61	1.42	1.35
8	2.05	2.35	2.22	1.98
9	1.64	1.94	1.78	1.47
Avg.	1.68	1.83	1.72	1.59

Table 10. AF4 decoder results for synthetic encoding, where only one point is activated.

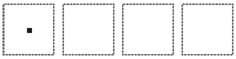
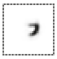
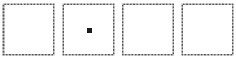
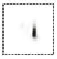
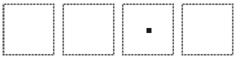
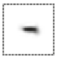
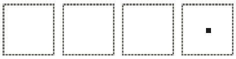
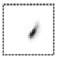
	Code	Output
#1		
#2		
#3		
#4		

Table 11. The sums of occurrence intensities (values in the encoding) for each visual feature AF4. It must be emphasized that the results in this table are relative, so they can only be compared to the numbers from the same table.

	#1	#2	#3	#4
0	4.89	5.12	3.20	5.25
1	3.90	4.19	2.14	3.63
2	5.07	5.20	3.36	5.77
3	5.18	5.44	3.58	5.08
4	5.03	5.72	2.89	5.07
5	4.92	4.97	3.26	4.76
6	5.38	5.00	3.17	4.92
7	4.78	5.14	2.88	4.33
8	5.41	5.72	3.46	5.27
9	5.08	5.00	3.21	4.65
Avg.	4.95	5.14	3.10	4.85

Table 12. AF5 decoder results for synthetic encoding, where only one point is activated.

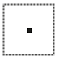
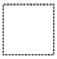
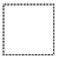
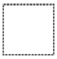
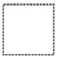
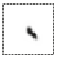
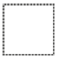
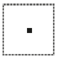
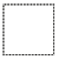
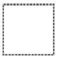
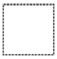
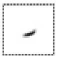
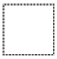
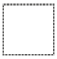
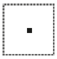
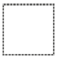
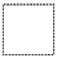
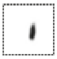
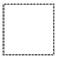
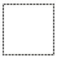
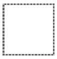
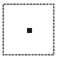
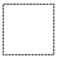
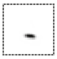
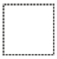
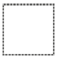
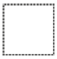
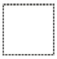
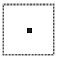
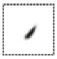
	Code					Output
#1						
#2						
#3						
#4						
#5						

Table 13. The sums of occurrence intensities (values in the encoding) for each visual feature AF5. It must be emphasized that the results in this table are relative, so they can only be compared to the numbers from the same table.

	#1	#2	#3	#4	#5
0	2.48	2.71	2.81	3.68	3.03
1	1.38	1.25	1.47	2.21	1.57
2	2.73	3.18	2.59	3.87	3.11
3	2.73	3.38	2.32	4.02	2.88
4	2.11	2.34	2.27	3.55	2.54
5	2.61	2.94	2.09	4.10	2.72
6	2.50	2.79	2.50	3.64	2.87
7	1.89	2.56	2.09	3.43	2.40
8	2.85	3.26	2.64	3.76	2.96
9	2.18	2.85	2.16	3.45	2.55
Avg.	2.33	2.70	2.28	3.55	2.65

The conclusions drawn from Tables 9, 11, and 13 can be further explained with proper illustrations. Decoding a single point did not provide a satisfactory understanding of the visual features (as was the case with feature #3 in AF5), which provides motivation to search for a better way of feature visualization. Instead, we can use the decoder for the actual multiple-pixel combinations generated for inputs from the test set. The sparse representation can be further split into separate matrices, related to different visual features. The decoding of a single matrix can be considered as partial reconstruction, which consists only of occurrences of the corresponding feature. For example, using only the first channel of the model with manually selected features will result in a partial reconstruction of a digit that consists of horizontal lines only. Additionally, in order to explain the limitations of presented methods, specific digits with especially low and especially high autoencoder errors were chosen for this visualization. Selected results for the discussed models are presented in Figures 8–10.

The rows described as encoding in Figures 8–13 contain the visualizations of tuples of sparse matrices. For the sake of readability, the active elements, which are naturally rare, were magnified threefold. The features row provides information on the distribution of particular types of visual features on the image plane. This may be associated with a particular digit segment that possesses that feature. It must be emphasized, however, that the decoder output cannot be described as a sum of separate features—the CNN-based decoder function is nonlinear and non-additive. The encoded information on the presence of a selected feature indicates not only the presence of specific digit segments associated with that feature, but it may also be indicative of the absence of other features, as is the case with digit 8 (Figure 8).

A similar approach was applied to larger images. In order to include even more information in the presented visualization, images containing both digits and other symbols were used. The results are shown in Figures 11–13. The autoencoders were trained in a digit-specific way, but the test images used in this case contain other symbols as well. The selected non-digit shapes, however, generated visible errors—some segments were erroneously enlarged, merged, broken into pieces or blurred.

An additional demonstration of the proposed method using a longer text fragment, which is a 512 × 128 scan of a postal address, is presented in Figures 14–16. They depict the encoding contents and decoder outputs for the three proposed models. The number of active elements in the sparse encoding is proportional to the image size, which is visibly larger than in the other examples. Digits are clearly readable in the decoder output, as their size is roughly similar to the MNIST samples. Some of the most significant errors occur for the pairs of letters that are especially close to each other, which are visible in the second line of text.

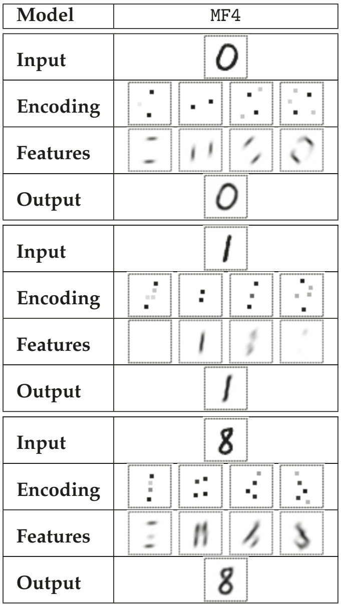


Figure 8. Sample encoding of selected digits performed by MF4 model. Apart from the input and output data and highly-sparse encoding (up to 5 non-zero elements in each encoding matrix), visualizations of single features are presented. Each visualization was acquired by decoding a synthetic code, where one of the visualized matrices was used, and all the other matrices were filled with zeros.

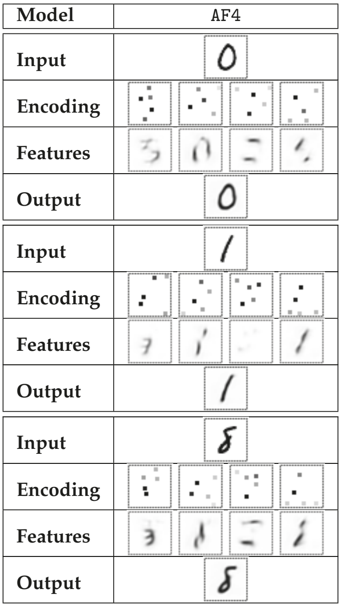


Figure 9. Sample encoding of selected digits performed by AF4 model. Highly sparse encoding (up to 5 non-zero elements in each encoding matrix) was used. Feature-specific decodings are presented, similar to those in Figure 8.

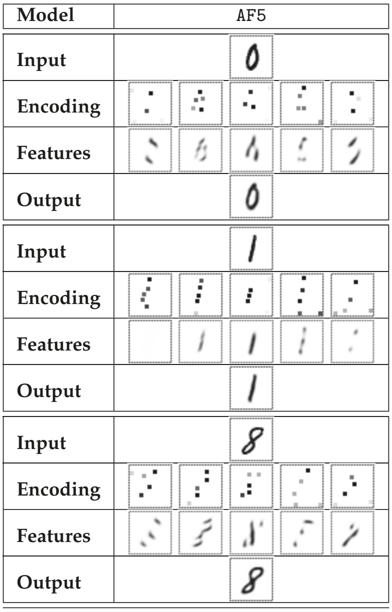


Figure 10. Sample encoding of selected digits performed by AF5 model. Highly sparse encoding (up to 5 non-zero elements in each encoding matrix) was used. Feature-specific decodings are presented, similar to those in Figure 8.

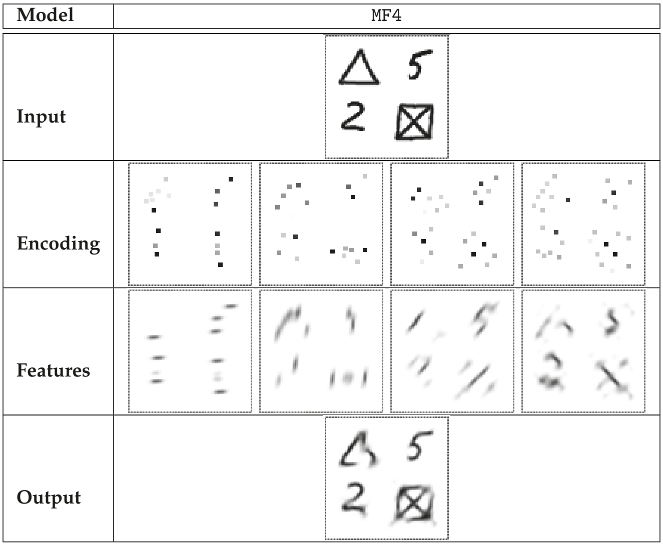


Figure 11. Sample encoding of a larger image that contains both digits and other symbols, performed with the MF4 model. The presented feature-specific decodings were generated in the same way as those presented in Figure 8.

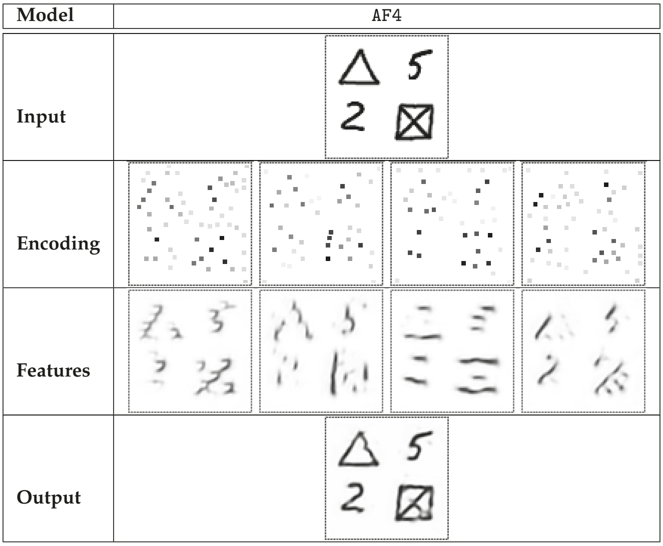


Figure 12. Sample encoding of a larger image that contains both digits and other symbols, performed with the AF4 model. The presented feature-specific decodings were generated in the same way as those presented in Figure 8.

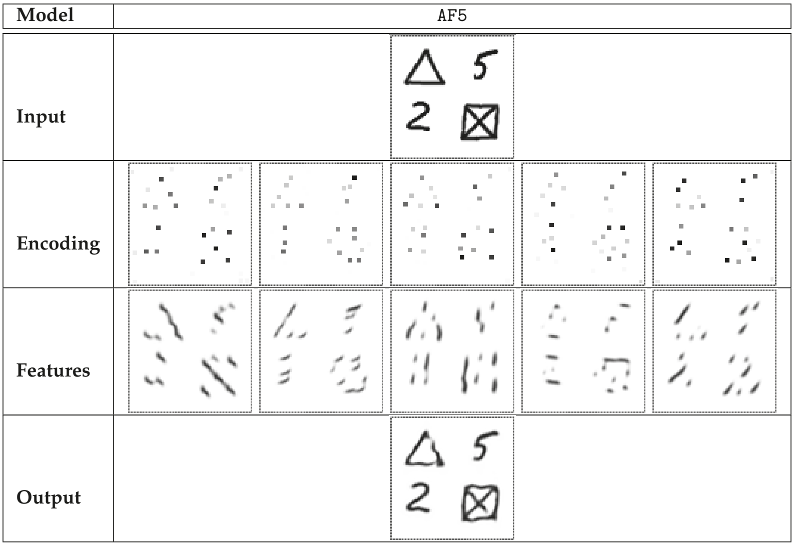


Figure 13. Sample encoding of a larger image that contains both digits and other symbols, performed with the `AF5` model. The presented feature-specific decodings were generated in the same way as those shown in Figure 8.

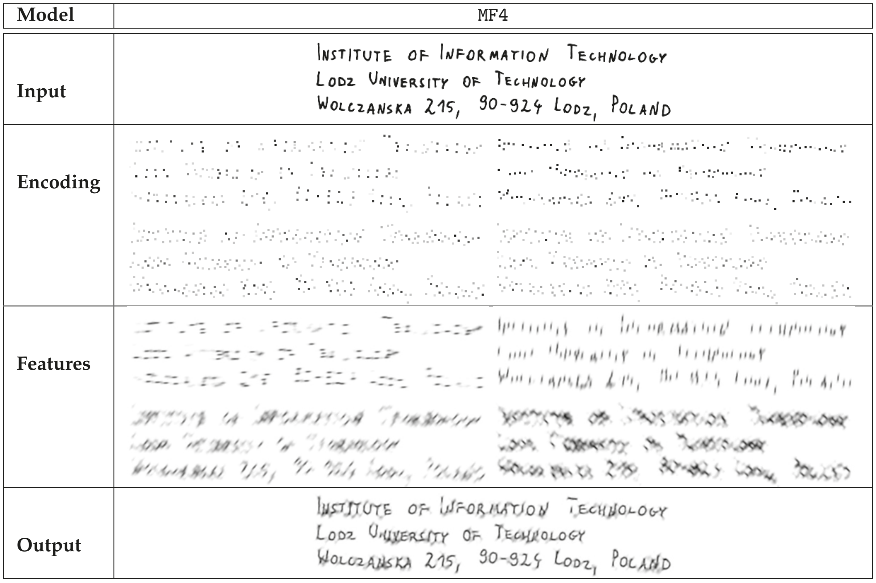

Model
MF4
Input
INSTITUTE OF INFORMATION TECHNOLOGY
LODZ UNIVERSITY OF TECHNOLOGY
WOLCZANSKA 215, 90-924 LODZ, POLAND
Encoding
Features
Output

Figure 14. Sample encoding of a postal address scan that contains both digits and letters, performed with the `MF4` model. The presented feature-specific decodings were generated in the same way as those shown in Figure 8.

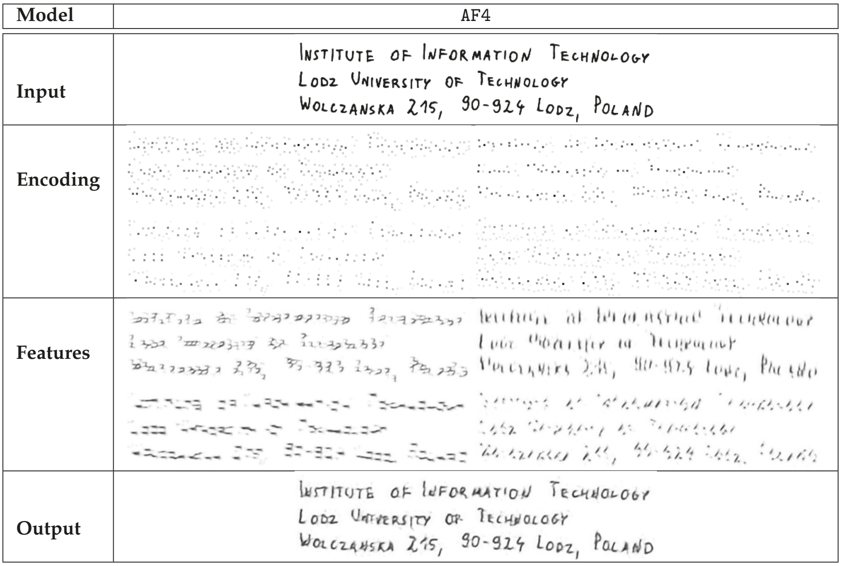

Model
AF4
Input
Encoding
Features
Output
INSTITUTE OF INFORMATION TECHNOLOGY
LODZ UNIVERSITY OF TECHNOLOGY
WOLCZANSKA 215, 90-924 LODZ, POLAND

Figure 15. Sample encoding of a postal address scan that contains both digits and letters, performed with the AF4 model. The presented feature-specific decodings were generated in the same way as those shown in Figure 8.

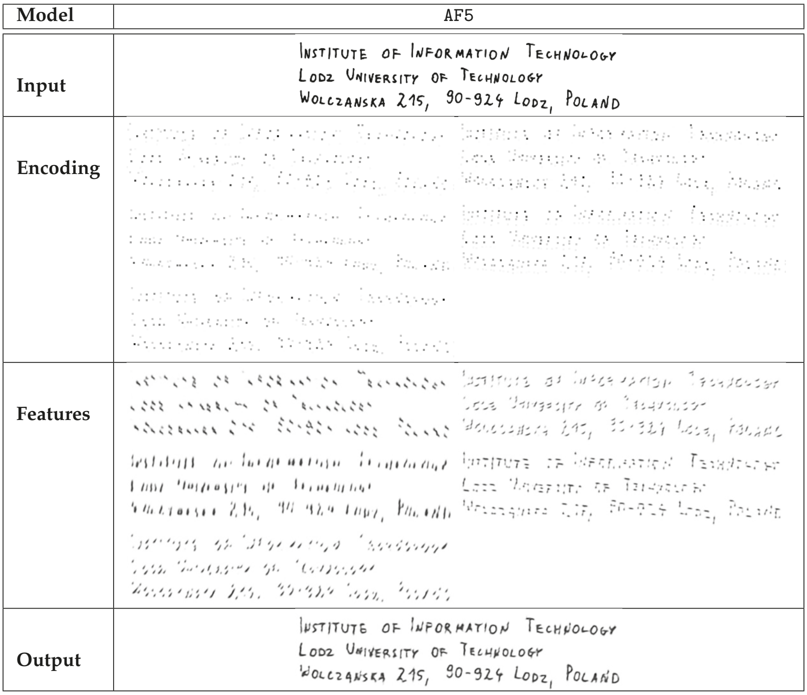

Model
AF5
Input
Encoding
Features
Output
INSTITUTE OF INFORMATION TECHNOLOGY
LODZ UNIVERSITY OF TECHNOLOGY
WOLCZANSKA 215, 90-924 LODZ, POLAND

Figure 16. Sample encoding of a postal address scan that contains both digits and letters, performed with the AF5 model. The presented feature-specific decodings were generated in the same way as those shown in Figure 8.

The presented illustrations involved 28 × 28 MNIST samples, 80 × 80 images with multiple symbols and 512 × 128 scan of a postal address. However, the method is scalable for any size of an input image. The execution time is proportional to the number of pixels, as the relations presented in the Figure 17 are roughly quadratic. The presented calculation time is very small, despite the standard GPU set-up being used for a single image. Processing multiple images of similar size in batches would reduce the average per-image processing time even further.

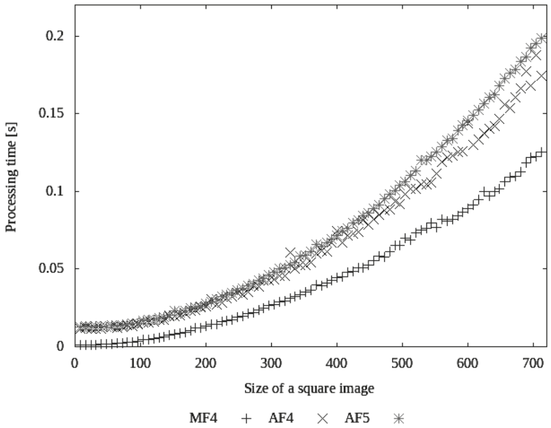


Figure 17. Processing times of `MF4`, `AF4`, and `AF5` models for square images of different sizes. As in the case of the standard training and test process, these results were achieved using GPU.

6. Conclusions

This paper has presented a novel method of image content representation. In our approach, we propose to encode an image as a tuple of sparse matrices that describes the intensity and position of selected visual features occurring in the image. The method was validated through a series of experiments on the MNIST data set [7]. The presented simple variant, in which each matrix was reduced to local maxima, provided the ability to generate sparse matrices where no more than 6.25% of elements were preserved. However, as revealed by further analysis, very sparse matrices with no more than five elements preserved in each 28 × 28 matrix (which is less than 0.64% of elements preserved) were sufficient to keep a low autoencoder error rate and obtain a classifier accuracy of 98.40%.

The application of the method to the classification task provided satisfactory results, outperforming the classical sparse coding solutions [23,25]. In our approach, the method of encoding was the same, regardless of the image class, because no class-dependent information was used in the training process. Thus, the presented method can be considered unsupervised. This is a relevant factor that has to be taken into account when comparing the results of the present study with those in [22]—our results are not as good as those of the supervised variant presented in that work but outperform those of the unsupervised variant.

The presented classifier also performs better than most solutions based on spiking neural networks [28,29]. It is not as good as some models presented in [31], but it must be emphasized that the classification accuracy comparison is not the key aspect in this particular case. Spiking neural networks possess different properties, such as the ability to handle noise [29]. Some of the models, such as LIF neurons [27,28] or STDP learning [29], suffer from too high complexity, or unsatisfactory sparsity level of the generated representation (especially in [31]).

The sparse representation based on visual features made it possible to perform a detailed analysis of the nature of the selected features. We presented both context-free feature visualizations (see Tables 8, 10, and 12) and digit-specific distribution of particular features (see Figures 8–13). Additionally, per-digit statistics of feature occurrences were discussed (see Tables 9, 11, and 13).

The presented method is based solely on convolutional layers and point-wise operations (activation functions). This makes the feature detection invariant to translation. Consequently, the pretrained autoencoder can process the input images of any size, which was demonstrated on the basis of a data set with larger images, each containing multiple digits (see Section 4.4). Experiments using non-digit symbols were also performed (see Figures 11–13), in order to show that the generated features were adjusted to the selected data set—not surprisingly, the autoencoder trained in the proposed way performed markedly worse on non-digit characters.

The study was based on the MNIST data set [7], which is relatively easy to analyze. However, further tests are needed to verify the applicability of the method to more complex databases, such as the ImageNet data set [37] or data from the Pascal Visual Object Classes Challenge [38]. In order to be applied to images of real-life objects, the method would have to be enhanced with the ability to recognize more complex and more numerous visual features.

The different visual features used in the presented approach are sensitive to object rotation and size. In particular, the images from the MNIST data set were easy to describe with lines, and the rotation of the line was the key element that identified the features. However, the task of feature detection in different rotations can be approached in multiple ways. Recent works on CNNs have provided new advances to transformation-invariant CNNs [14,39,40]. The prospect of combining those methods with the proposed representation, resulting in additional information about orientation and scale of detected keypoints, is another promising option worth pursuing for further development.

Author Contributions: Conceptualization, A.T., P.T., and B.S.; methodology, P.T. and A.T.; software, P.T.; validation, P.T.; formal analysis, P.T.; investigation, P.T., A.T., and B.S.; resources, P.T. and A.T.; data curation, P.T.; writing—original draft preparation, P.T, A.T., and B.S.; writing—review and editing, P.T., A.T., and B.S.; visualization, P.T.; supervision, A.T.; project administration, A.T.; funding acquisition, A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This project has been partly funded with support from the National Science Centre, Republic of Poland, Decision Number DEC-2012/05/D/ST6/03091.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Marr, D.; Ullman, S.; Poggio, T. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*; The MIT Press: Cambridge, MA, USA, 2010.
2. Lecun, Y.; Bengio, Y. Convolutional Networks for Images, Speech and Time Series. In *The Handbook of Brain Theory and Neural Networks*; Arbib, M.A., Ed.; The MIT Press: Cambridge, MA, USA, 1995; pp. 255–258.
3. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerlan, 2014; pp. 818–833. [CrossRef]
4. Gonzalez-Garcia, A.; Modolo, D.; Ferrari, V. Do Semantic Parts Emerge in Convolutional Neural Networks? *Int. J. Comput. Vis.* **2018**, *126*, 476–494. [CrossRef]
5. Simon, M.; Rodner, E.; Denzler, J. Part Detector Discovery in Deep Convolutional Neural Networks. In *Computer Vision, Proceedings of the ACCV 2014—12th Asian Conference on Computer Vision, Singapore, Singapore, 1–5 November 2014, Revised Selected Papers, Part II*; Lecture Notes in Computer Science; Cremers, D., Reid, I.D., Saito, H., Yang, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 9004, pp. 162–177. [CrossRef]
6. Zhang, Q.; Wang, X.; Cao, R.; Wu, Y.N.; Shi, F.; Zhu, S. Explanatory Graphs for CNNs. *CoRR* **2018**, *abs/1812.07997*.
7. LeCun, Y.; Cortes, C. MNIST handwritten digit database.
8. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Susstrunk, S. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]

9. von Gioi, R.G.; Jakubowicz, J.; Morel, J.M.; Randall, G. LSD: a Line Segment Detector. *Image Process. Line* **2012**, *2*, 35–55. [CrossRef]
10. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
11. Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.* **2008**, *110*, 346–359. [CrossRef]
12. Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P. Geometric Deep Learning: Going beyond Euclidean Data. *IEEE Signal Process. Mag.* **2017**, *34*, 18–42. [CrossRef]
13. Monti, F.; Boscaini, D.; Masci, J.; Rodolà, E.; Svoboda, J.; Bronstein, M.M. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 5425–5434.
14. Tomczyk, A.; Szczepaniak, P.S. Ear Detection Using Convolutional Neural Network on Graphs with Filter Rotation. *Sensors* **2019**, *19*, 5510. [CrossRef]
15. Tomczyk, A. Active Partitions in Localization of Semantically Important Image Structures. In *Bridging the Semantic Gap in Image and Video Analysis*; Intelligent Systems Reference Library; Kwaśnicka, H., Jain, L.C., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 51–72.[CrossRef]
16. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning Deep Features for Discriminative Localization. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2921–2929.
17. Liu, B.; Wang, M.; Foroosh, H.; Tappen, M.; Penksy, M. Sparse Convolutional Neural Networks. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 806–814. [CrossRef]
18. Gu, S.; Zuo, W.; Xie, Q.; Meng, D.; Feng, X.; Zhang, L. Convolutional Sparse Coding for Image Super-Resolution. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 5–13 December 2015; pp. 1823–1831. [CrossRef]
19. Sorel, M.; Sroubek, F. Fast convolutional sparse coding using matrix inversion lemma. *Digital Signal Processing* **2016**, *55*, 44–51. [CrossRef]
20. Heide, F.; Heidrich, W.; Wetzstein, G. Fast and flexible convolutional sparse coding. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 5135–5143. [CrossRef]
21. Yang, M.; Zhang, L.; Feng, X.; Zhang, D. Sparse Representation Based Fisher Discrimination Dictionary Learning for Image Classification. *Int. J. Comput. Vis.* **2014**, *109*, 209–232. [CrossRef]
22. Yang, J.; Yu, K.; Huang, T. Supervised translation-invariant sparse coding. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3517–3524. [CrossRef]
23. Chen, B.; Li, J.; Ma, B.; Wei, G. Convolutional sparse coding classification model for image classification. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), 25–28 September 2016; pp. 1918–1922. [CrossRef]
24. Zhang, Z.; Li, F.; Chow, T.W.S.; Zhang, L.; Yan, S. Sparse Codes Auto-Extractor for Classification: A Joint Embedding and Dictionary Learning Framework for Representation. *IEEE Trans. Signal Process.* **2016**, *64*, 3790–3805. [CrossRef]
25. Wang, Z.; Yang, J.; Nasrabadi, N.; Huang, T. A Max-Margin Perspective on Sparse Representation-Based Classification. In Proceedings of the 2013 IEEE International Conference on Computer Vision, 1–8 December 2013; pp. 1217–1224. [CrossRef]
26. Cao, Y.; Chen, Y.; Khosla, D. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *Int. J. Comput. Vis.* **2015**, *113*, 54–66. [CrossRef]
27. Bugmann, G.; Christodoulou, C.; Taylor, J.G. Role of Temporal Integration and Fluctuation Detection in the Highly Irregular Firing of a Leaky Integrator Neuron Model with Partial Reset. *Neural Comput.* **1997**, *9*, 985–1000. [CrossRef]
28. Hunsberger, E.; Eliasmith, C. Spiking Deep Networks with LIF Neurons. *CoRR* **2015**, *abs/1510.08829*.
29. Tavanaei, A.; Maida, A.S. Bio-Inspired Spiking Convolutional Neural Network using Layer-wise Sparse Coding and STDP Learning. *CoRR* **2016**, *abs/1611.03000*.

30. Pearson, K. Skew variation, a rejoinder. *Biometrika* **1905**, *4*, 169–212.
31. Diehl, P.U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.C.; Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8.
32. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional Architecture for Fast Feature Embedding. In Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, 3–7 November 2014; ACM: New York, NY, USA, 2014; pp. 675–678. [CrossRef]
33. Hubel, D.H.; Wiesel, T.N. Receptive Fields and Functional Architecture in Two Nonstriate Visual Areas (18 and 19) of the Cat. *J. Neurophysiol.* **1965**, *28*, 229–289. [CrossRef]
34. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, Santiago, Chile, 7–13 December 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 1026–1034. [CrossRef]
36. Wan, L.; Zeiler, M.; Zhang, S.; LeCun, Y.; Fergus, R. Regularization of Neural Networks Using Dropconnect. In Proceedings of the 30th International Conference on International Conference on Machine Learning—Volume 28. JMLR.org, 2013, ICML'13, Atlanta, GE, USA, 17–19 June 2013; p. III-1058–III-1066.
37. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR) 2009, Miami, FL, USA , 20–25 June 2009.
38. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
39. Laptev, D.; Savinov, N.; Buhmann, J.M.; Pollefeys, M. TI-POOLING: transformation-invariant pooling for feature learning in Convolutional Neural Networks. *CoRR* **2016**, *abs/1604.06318*.
40. Tarasiuk, P.; Pryczek, M. Geometric Transformations Embedded into Convolutional Neural Networks. *J. Appl. Comput. Sci. (JACS)* **2016**, *24*, 33 – 48.

applied sciences

MDPI

Article

Fast Self-Adaptive Digital Camouflage Design Method Based on Deep Learning

Houdi Xiao, Zhipeng Qu *, Mingyun Lv, Yi Jiang, Chuanzhi Wang and Ruiru Qin

School of Aeronautic Science and Engineering, Beihang University, No. 37 Xueyuan Road, Haidian District, Beijing 100191, China; xhdbuaa@buaa.edu.cn (H.X.); lv503@buaa.edu.cn (M.L.); jiangyi312@buaa.edu.cn (Y.J.); wangchuanzhi@buaa.edu.cn (C.W.); qrraxsy@buaa.edu.cn (R.Q.)

* Correspondence: quzhipeng@buaa.edu.cn

Received: 20 June 2020; Accepted: 28 July 2020; Published: 30 July 2020

Abstract: Traditional digital camouflage is mainly designed for a single background and state. Its camouflage performance is appealing in the specified time and place, but with the change of place, season, and time, its camouflage performance is greatly weakened. Therefore, camouflage technology, which can change with the environment in real-time, is the inevitable development direction of the military camouflage field in the future. In this paper, a fast-self-adaptive digital camouflage design method based on deep learning is proposed for the new generation of adaptive optical camouflage. Firstly, we trained a YOLOv3 model that could identify four typical military targets with mean average precision (mAP) of 91.55%. Secondly, a pre-trained deepfillv1 model was used to design the preliminary camouflage texture. Finally, the preliminary camouflage texture was standardized by the k-means algorithm. The experimental results show that the camouflage pattern designed by our proposed method is consistent with the background in texture and semantics, and has excellent camouflage performance in optical camouflage. Meanwhile, the whole pattern generation process takes a short time, less than 0.4 s, which meets the camouflage design requirements of the near-real-time camouflage in the future.

Keywords: adaptive camouflage; convolutional neural network (CNN); k-means; object detection; image completion; machine learning; saliency detection

1. Introduction

Camouflage is the most common and effective means to combat military reconnaissance [1,2]. It can conceal military equipment in natural environments. With the development of camouflage technology, optical camouflage has evolved from deformable camouflage to digital camouflage [3]. However, traditional digital camouflage is mainly designed for a single background and state [4]. In the traditional digital camouflage design method, the colors are only the main color of a specific environment, and the texture is formed by the non-random algorithm arrangement of finite pattern templates [5]. The traditional way of realizing digital camouflage is to coat the equipment surface with camouflage paint according to the designed camouflage texture or to wear or cover the fabric with camouflage texture. There is a limitation that the camouflage performance of a specified time and place is appealing, and the camouflage performance is greatly weakened when the location, season, and time changes. Cross-region, multi-season, multi-period camouflage has become a new military demand for modern weapons and equipment. Therefore, the camouflage technology, which can change with the environment in real-time has become the inevitable direction of the development of the military camouflage field. During the last decades, much effort has been directed toward achieving this goal. To realize multi-region adaptive camouflage, texture and color must not be fixed, real-time camouflage texture is designed according to the changes in the environment. Different from the traditional camouflage, the implementation way needs to use a controllable multi-color variable material. By fabricating the controllable color-changing material into color-changing units embedded on the surface of the camouflaged target, just like the cells that make up the chameleon's skin, we can refer to the whole device as camouflage skin. The external control system controls the camouflage skin to display the design's camouflage texture.

Researchers find inspiration from nature. It is well known that there are loads of animals in nature with excellent camouflage abilities, such as cephalopods (like squid and cuttlefish) [6–9], chameleons [10], some insects [11], and so on. These animals have amazing control over their appearance (such as color, contrast, pattern). The principle of animal camouflage is to use the nervous system to sense changes in the environment and control the cells on the skin to change into different colors and textures according to the environment. Inspired by the principle of animal camouflage, researchers have designed various types of color-changing devices or camouflage samples [12–14], to mimic cells on the surface of the animal's skin. Single material for all colors has been reported [15–17]; it is an inverse polymer-gel opal which is prepared from an electroactive material. It can be stimulated by an electric field to change colors. In addition, devices that use magnetic field stimulation to achieve color changes have also been reported [18]. A mechanical chameleon based on dynamic plasmonic-nanostructures has been designed [14]. At present, most researchers focus on the technology of controllable color change, but there are few reports on the design method of new generation self-adaptive camouflage texture [19].

As one of the key technologies of adaptive optical camouflage, the study of adaptive camouflage texture design has important theoretical and practical significance. In this paper, the design method of adaptive camouflage texture for typical military targets in the natural environment is studied. With the development of computer vision technology, deep learning has been applied to various image processing scenarios. It has been used for image classification [20–22], object detection [23–25], image segmentation [25–28], image completion [29–31], and so on, and has achieved a series of amazing results. However, there are few reports on the application of the current achievements of deep learning to the field of military camouflage. We wondered if deep learning could be used to mimic the way that the animal's nervous system senses the environment and controls skin cells to change color and texture. Hence, we proposed a fast-self-adaptive digital camouflage design method based on deep learning. The method can realize the recognition of camouflage target and the design of adaptive camouflage pattern in near real-time. Firstly, we used the YOLOv3 algorithm to realize the recognition of typical military targets. Secondly, we used the deepfillv1 algorithm to realize the preliminary design of adaptive camouflage texture. Finally, the k-means algorithm was used to realize the standardization of adaptive camouflage texture. The experimental results showed that the camouflage pattern designed by our proposed method is consistent with the background in texture and semantics. It has excellent camouflage performance in optical camouflage. The whole process took less than 0.4 s. All experiments are implemented on Python3.6, TensorFlow v1.6, CUDNN v7.1, CUDA v9.2, and run on hardware with a CPU Intel Core I7-9700F (3.0 GHz) and GPU RTX 2080 Ti. The proposed camouflage pattern design method has potential application value in future real-time optical camouflage.

2. Literature Review

Military camouflage colors and patterns have evolved throughout history to improve their effectiveness, with each variant designed for a specific environment. Therefore, camouflage patterns are only effective in areas where the local background remains relatively constant. For a military system to operate in a variety of environments, its camouflage must be adjusted accordingly [32]. As a result, researchers around the world are beginning to design adaptive camouflage techniques that can change the color and texture of surfaces, like chameleons or octopuses, depending on their environment. Some researchers propose to project the collected background image on the surface of the target to achieve the purpose of camouflage. Inami et al. [33,34] designed an active camouflage system. The system first obtains the real-time background image through the image acquisition device installed on the back of the target and then projects the display scheme calculated from the observer's perspective onto the target surface with reflective materials. Morin et al. [13] used microfluid network technology to prepare a soft camouflaging robot. It could change the color, contrast, pattern, apparent shape, luminescence, and surface temperature. The researchers changed the robot's color, pattern, and so on by filling the tiny tubes with different colored liquids. Pezeshkian et al. [32] proposed the use of gray-level co-occurrence matrices to synthesize a texture similar to the background. Then, the texture is displayed on the surface of the battlefield reconnaissance robot by electronic paper display technology to achieve the purpose of camouflage. Inspired by the skin discoloration principle of cephalopods, Yu et al. [35] used a thermochromic material to prepare a photoelectric camouflage demonstration system that could transform between black and white. The color-changing material is colorless and transparent when the temperature is higher than 47 °C, and black when the temperature is lower than 47 °C. By controlling the temperature of each unit, the researchers can display different patterns. Unfortunately, only black and white patterns can be displayed. Wang et al. [14] used the adjustable plasmon technology to prepare a color-changing device that could cover the whole visible band and then developed a bionic mechanical chameleon. The mechanical chameleon could sense color changes in its environment and actively change its own color to match the color of its environment. It is a pity that the author did not study the design method of camouflage texture. So far, researchers have focused on how to design and implement color change, but few have studied how to design appropriate adaptive camouflage textures. The existing researches on camouflage texture mainly focus on traditional camouflage texture. For example, Xue et al. [5] designed digital camouflage textures based on recursive overlapping of pattern templates. Zhang et al. [36] proposed a digital camouflage design method based on a space color mixing model. The model can simulate the color-mixing process in the aspects of color-mixing order, shape, and position of color-mixing spot. Jia et al. [37] proposed a camouflage pattern design method based on spot combination. The core idea of the above design method is random or non-random arrangement of finite templates. Due to the simplicity of the template, these traditional design methods cannot meet the needs of the new generation of adaptive camouflage texture design. Therefore, it is necessary to study the design method of adaptive camouflage texture.

3. Methodology

3.1. Outline of Proposed Method

The essence of optical camouflage is to make it impossible for human eyes or optical cameras to distinguish a target from its environment. This is similar to target removal in image processing. Inspired by this, we applied the image completion algorithm to the design of camouflage texture. In this paper, we provide a quick method based on the convolutional neural network to generate adaptive digital camouflage. The flowchart of our method is shown in Figure 1. To achieve camouflage of the target, we need to identify the target that needs camouflage. First of all, we used the YOLOv3 [38] algorithm to conduct recognition model training for four typical military targets. After adjusting the hyper parameters, we obtained a model with good recognition probability. Secondly, we masked the target area and entered it into the deepfillv1 [39] model that was pre-trained by places2 [40] for image completion. In this step, we obtain the preliminary camouflage texture. Thirdly, we used the k-means algorithm to calculate the main color of the filled area and compared it with the military standard color. The most similar color in the standard was selected as the main color, and the corresponding color was replaced. At this point, we have the final adaptive camouflage texture. The digital camouflage generated by this method is consistent with the texture of the surrounding environment. This method can generate visually plausible camouflage pattern structures and textures.

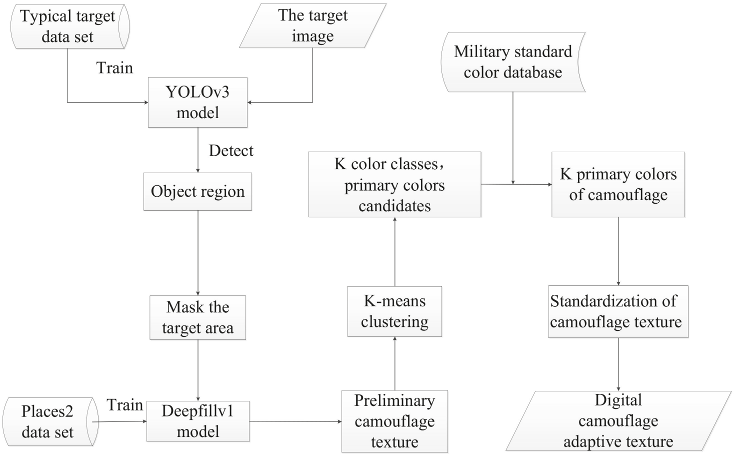

Typical target data set
Train
The target image
YOLOv3 model
Military standard color database
Detect
Object region
K color classes，primary colors candidates
K primary colors of camouflage
Mask the target area
K-means clustering
Standardization of camouflage texture
Places2 data set
Train
Deepfillv1 model
Preliminary camouflage texture
Digital camouflage adaptive texture

Figure 1. Outline of proposed digital camouflage design method.

3.2. Dataset

In this paper, the Imagenet2012 [41] and Places2 [40] data sets were used. The Imagenet2012 [41] dataset consists of over 1.28 million images, containing 1000 categories, with the number of images per category ranging from 732 to 1300. Four typical military target images were selected from the ImageNet2012 [41]. This dataset was segmented into a training and test set. The four typical targets are airships, aircraft carriers, tanks, and uniformed soldiers. A total of 2187 images were selected from the dataset, of which 1970 were used for training and 217 for testing. There are no less than 500 pictures in each category. Table 1 shows the number of train and test images for the different categories. Figure 2 shows one sample for airships, aircraft carriers, tanks, and uniformed soldier images, which was selected from Imagenet2012.

Figure 2. Data samples from the Imagenet2012 dataset. (**a**) airship; (**b**) aircraft carrier; (**c**) tank; (**d**) uniformed soldier.

The Place2 [35] dataset contains more than 400 different types of scene environments and 10 million images. Basically, covering people’s common scenes. Figure 3 shows one sample for forest, desert, grassland, and snowfield environment images, which was selected from Places2.

Table 1. Details of the training and test set.

Category	Training Set	Test Set
Airships	459	50
Aircraft Carriers	455	50
Tanks	496	55
Uniformed Soldiers	560	62
Total	1970	217

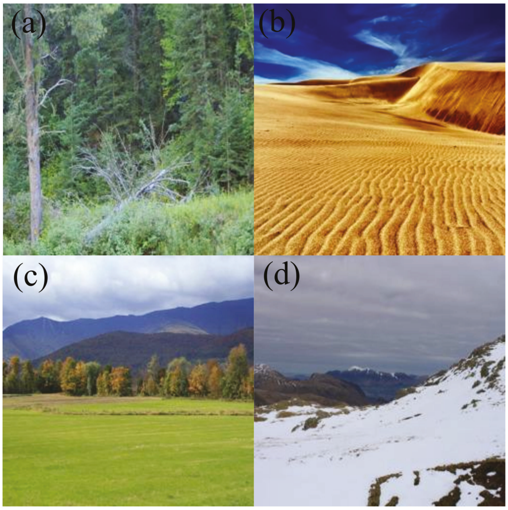


Figure 3. Data samples from the Places2 dataset. (**a**) forest; (**b**) desert; (**c**) grassland; (**d**) snowfield.

3.3. Military Target Detection Based on YOLOv3

To achieve camouflage of the target, we first needed to identify the target that needs camouflage. The YOLOv3 [38] algorithm was used to identify military targets. The Yolo series algorithm is an algorithm that could detect objects quickly [38,42,43]. The YOLOv3 is the latest version [38]. YOLOv3 can achieve high precision real-time detection, which is very suitable for our application background. Its network structure is shown in Figure 4. The resolution of the input picture in the network structure diagram is 416 × 416 × 3 (in fact, it can be any resolution.), and has four labeled classes. It uses darknet-53, which removes the full connection layer, as the backbone network. The YOLOv3 is a fully convoluted network that makes extensive use of residual network structures. As shown in Figure 4, YOLOv3 consists of DBL, resn, Up-sample, and concat. DBL stands for convolution (conv), batch normalization (BN) and leaky relu activation (Leaky reu). Resn represents the *n* residual units (res unit) in this residual block (res_block). Zero padding means using zero to fill the edge of the image. Up-sampling represents up-sampling. The concat represents the merging tensor. DBL*n represents the *n* DBL. The add represents the addition operation. The following y1, y2, and y3 represent feature maps with three different dimensions.

The network structure of darknet-53 is shown in Table 2. It uses successive 3 × 3 and 1 × 1 convolutional layers and some shortcut connections. The application of the shortcut connection layer allows the network to be deeper. It has 53 convolutional layers [38].

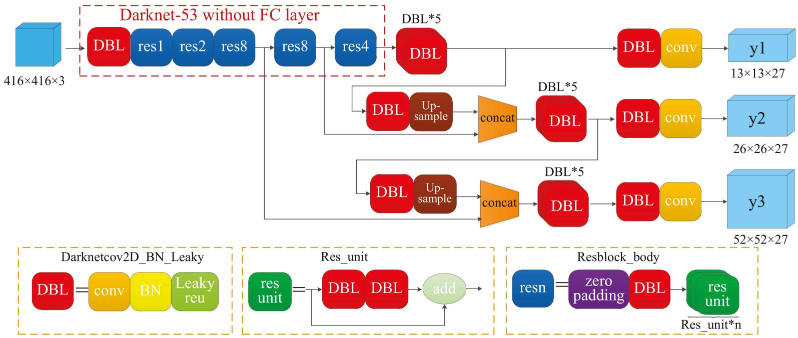


Figure 4. YOLOv3 network structure.

Table 2. Darknet-53 [38].

	Type	Filters	Size/Stride	Output
	Convolutional	32	3 × 3	416 × 416
	Convolutional	64	3 × 3/2	208 × 208
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			208 × 208
	Convolutional	128	3 × 3/2	104 × 104
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			104 × 104
	Convolutional	256	3 × 3/2	52 × 52
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			52 × 52
	Convolutional	512	3 × 3/2	26 × 26
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			26 × 26
	Convolutional	1024	3 × 3/2	13 × 13
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			13 × 13
	Avgpool		Global	
	Connected		1000	
	Softmax			

The loss function of YOLOv3 consists of localization loss $Loss_l$, confidence loss $Loss_c$, and classification loss $Loss_p$. The loss function is as follows:

$$Loss = Loss_l + Loss_c + Loss_p \tag{1}$$

$$Loss_l = \begin{array}{c} \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(x_i^j - \hat{x}_i^j \right)^2 + \left(y_i^j - \hat{y}_i^j \right)^2 \right] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(\sqrt{w_i^j} - \sqrt{\hat{w}_i^j} \right)^2 + \left(\sqrt{h_i^j} - \sqrt{\hat{h}_i^j} \right)^2 \right] \end{array} \tag{2}$$

$$Loss_c = \begin{array}{c} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i^j \log\left(C_i^j\right) + \left(1 - \hat{C}_i^j\right) \log\left(1 - C_i^j\right) \right] + \\ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} \left[\hat{C}_i^j \log\left(C_i^j\right) + \left(1 - \hat{C}_i^j\right) \log\left(1 - C_i^j\right) \right] \end{array} \tag{3}$$

$$Loss_p = \sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in calses} \left[\hat{p}_i^j(c) \log\left(p_i^j(c)\right) + \left(1 - \hat{p}_i^j(c)\right) \log\left(1 - p_i^j(c)\right) \right] \tag{4}$$

where $\lambda_{coord} = 5$, $I_{\mathrm{ij}}^{obj} = 1$ when the j th boundary box in cell i is responsible for detecting the object, otherwise 0. x, y, w, h denotes the bounding box parameter, C_i^j is the box confidence score of the box j in cell i, $\lambda_{noobj} = 0.5$, I_{ij}^{noobj} is the complement of I_{ij}^{obj}, $p_i^j(c)$ denotes the conditional class probability of the box j th in cell i for class c. All the letters with superscript represent the corresponding ground truth (GT) values.

The learning rate adopts cosine attenuation:

$$\alpha_{decayed} = \alpha_{end} + 0.5 * (\alpha_{initial} - \alpha_{end}) * (1 + \cos(s_{global}/s_{train} * pi)) \tag{5}$$

where $\alpha_{decayed}$ denotes the decayed learning rate; $\alpha_{initial}$ denotes the initial learning rate; α_{end} denotes the end learning rate; s_{train} denotes the total train steps; s_{global} denotes the global steps.

More details about YOlOv3 can be found in reference [38]. The basic code is online at https://github.com/YunYang1994/tensorflow-yolov3. Thanks to Yun Yang for sharing the code.

3.4. Preliminary Camouflage Texture Design Based on Deepfillv1

The essence of optical camouflage is to make it impossible for human eyes or optical cameras to distinguish a target from its environment. This is similar to target removal in image processing. Inspired by this, we applied the image completion algorithm to the design of camouflage texture. This method could be used to design the camouflage pattern consistent with the real-time background texture.

Deepfillv1 is a generated image inpainting model based on the contextual attention mechanism [39]. It can quickly generate a novel image structure consistent with the surrounding environment. The framework of deepfillv1 is shown in Figure 5. Deepfillv1 consists of two stages. The first stage is a simple dilated convolutional network trained with reconstruction loss to rough out the missing contents. The second stage is the training of the contextual attention layer. The core idea is to use the features of known image patches as the convolution kernel to process the generated patches to refine the fuzzy repair results. It is designed and implemented with convolution for matching generated patches with known contextual patches, channel-wise softmax to weigh relevant patches, and deconvolution to reconstruct the generated patches with contextual patches. The spatial propagation layer is used to improve the spatial consistency of the attention module. In order to make the network produce novel contents, another convolution path parallel to the contextual attention path is designed. The two paths are combined and fed to a single decoder for the final output. The entire network is trained end-to-end. The coarse network is trained explicitly with the reconstruction loss, while the refinement network is trained with the reconstruction, as well as global and local gradient penalty wasserstein GAN (WGAN-GP) [44,45] losses. The reconstruction loss uses a weighted sum of pixel-wise l_1 loss. The weight of each pixel is computed as γ^l, where l is the distance of the pixel to the nearest known pixel. γ is set to 0.99. The WGAN-GP uses the Earth-Mover distance $W\left(P_r, P_g\right)$ for comparing the generated and real data distributions. Its objective function is constructed by applying the Kantorovich–Rubinstein duality:

$$\min_G \max_D E_{\mathbf{x} \sim P_r}[D(\mathbf{x})] - E_{\tilde{\mathbf{x}} \sim P_g}\left[D\left(\tilde{\mathbf{x}}\right)\right] \tag{6}$$

where D is the set of 1-Lipschitz functions and P_g is the model distribution implicitly defined by $\tilde{\mathbf{x}} = \mathrm{G}(\mathbf{z})$. $\mathbf{z}$ is the input to the generator.

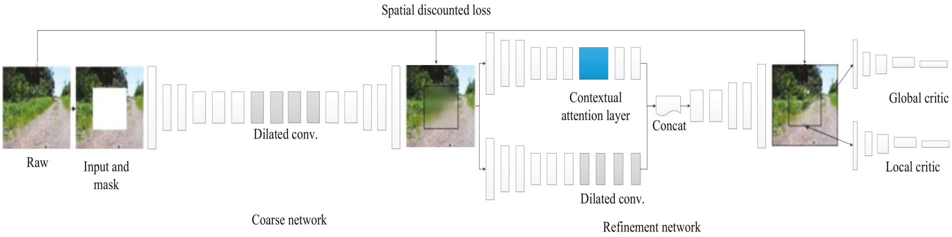


Figure 5. The framework of deepfillv1.

The $W\left(P_r, P_g\right)$ is as follows:

$$W\left(P_r, P_g\right) = \inf_{\gamma \in \prod\left(P_r, P_g\right)} E_{(\mathbf{x},\mathbf{y})\sim\gamma}\left[\left\|\mathbf{x} - \mathbf{y}\right\|\right] \tag{7}$$

where $\prod\left(P_r, P_g\right)$ denotes the set of all joint distributions $\gamma(\mathbf{x}, \mathbf{y})$ whose marginals are P_r and P_g, respectively.

More details about deepfillv1 can be found in reference [39]. The basic code is online at https://github.com/JiahuiYu/generative_inpainting. Thanks to Jiahui Yu for sharing the code.

3.5. Standardization of Camouflage Texture based on K-means

The initial camouflage texture generated previously, although visually well integrated into the background, cannot be directly applied to the actual camouflage due to its large amount of colors. This is partly because it contains too many colors, which makes it difficult to operate in practical projects. On the other hand, the patterns generated by this method change with the environment, which leads to a huge increase in color further. Therefore, it is necessary to choose a limited number of colors as representative colors according to certain standards to replace similar colors, so as to achieve a balance between camouflage performance and engineering practice. We call this process the standardization of camouflage texture.

Based on the traditional digital camouflage color extraction method, we used the k-means clustering algorithm to extract the main color of the camouflage area. Note that extracting the primary color region here is different from the traditional method. We extract the preliminary camouflage designed area, while the traditional method is to extract the whole background. The flowchart of the extraction process of primary colors is shown in Figure 6. The extracted primary color also needs to meet the following restrictions [5]:

1. The primary colors should have different brightness so that the camouflage pattern could destroy the shape of the camouflaged target.
2. The primary colors should not be too different from the background colors.

The Red-green-blue (RGB), hue-saturation-value (HSV), and Lab color spaces are commonly used in image processing. The RGB color space is related to devices, it does not reflect the true nature of human vision. However, the Lab model is a device-independent color system and based on physiological characteristics. This means that it is a digital way to describe human vision. In this paper, we chose the Lab color space since it can mimic the human vision system more closely.

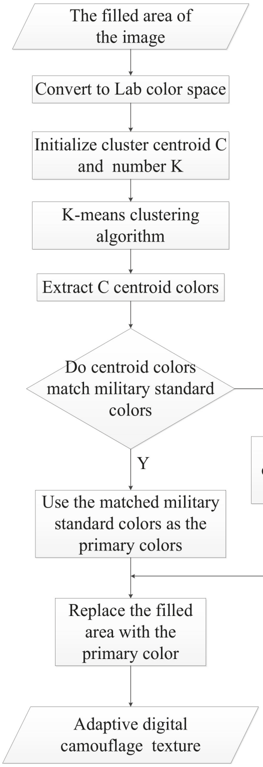

The filled area of the image
Convert to Lab color space
Initialize cluster centroid C and number K
K-means clustering algorithm
Extract C centroid colors
Do centroid colors match military standard colors
N
Y
Use the nearest K colors as the primary colors
Use the matched military standard colors as the primary colors
Replace the filled area with the primary color
Adaptive digital camouflage texture

Figure 6. Standardization of camouflage texture4 results.

Figure 6 shows the standardization process for camouflage textures. Firstly, we converted the color space of the filled area from RGB space to Lab space. Secondly, we needed to initialize cluster centroid C and number K. Normally, C is set randomly, and K is set to 4 or 5. Thirdly, pixels in the filled area of the image were classified into different categories according to their distance from the clustering centroids. Then, the most representative color in each pixel category was selected as the representative color of each category. According to the geographical environment of the background, digital camouflage is usually divided into four types—woodland, desert, ocean, and urban camouflage. According to a large number of data and actual production experience, the standard primary colors of various digital camouflage are determined. In this study, after determining the representative colors of the target area, we selected the standard color, which is closest to the representative color as the primary colors for designing the target camouflage. Finally, we used the standard color to replace the color of the pixel in the filled area to obtain the self-adaptive digital camouflage texture.

We used the Euclidean distance to calculate the distance between the representative colors and the standard colors. The distance of one color pair $d(r,s)$ is computed as:

$$d(r,s) = \sqrt{(L_r - L_s)^2 + (a_r - a_s)^2 + (b_r - b_s)^2} \tag{8}$$

4. Results

4.1. Military Target Detection

We used the method described in Section 3.3 to detect typical military targets. Four typical military target images were selected from the ImageNet2012 [41]. This dataset was segmented into a training and testing set. The four typical targets are airships, aircraft carriers, tanks, and uniformed soldiers. A total of 2187 images were selected from the dataset, of which 1970 were used for training and 217 for testing. There were no less than 500 pictures in each category. Unless otherwise noted, all the original images in this article about the four typical targets were from Imagenet2012.

The initial training parameters are shown in Table 3. $IOU_{threshold}$ is the intersection over union (IOU) threshold. We used multi-scale training methods.

Table 3. The training parameters.

Variable	Value	Variable	Value
$IOU_{threshold}$	0.5	$\alpha_{initial}$	1×10^{-4}
Batch_size	6	α_{end}	1×10^{-6}
Input_size	[320, 352, 384, 416, 448, 480, 512, 544, 576, 608]	α_{train}	

We used k-means clustering to determine our nine bounding box priors. On the selected dataset, the nine clusters were: (55 × 69), (151 × 91), (84 × 261), (200 × 188), (331 × 137), (179 × 346), (358 × 223), (350 × 303), (373 × 387).

We used the pre-training weight on the coco data set as the initialization weight. After 17 k steps of training, the model converged. The training loss is shown in Figure 7. As shown in Figure 7a, after 17 k steps of training, the loss was reduced to 0.6, which is basically convergent. The mean average precision (mAP) at IOU = 0.5 (mAP@.5) was 91.55%, as shown in Figure 7b. The results showed that the model we trained had high precision and could meet our application requirements. The total loss was calculated on the training set, and the mAP was calculated on the test set.

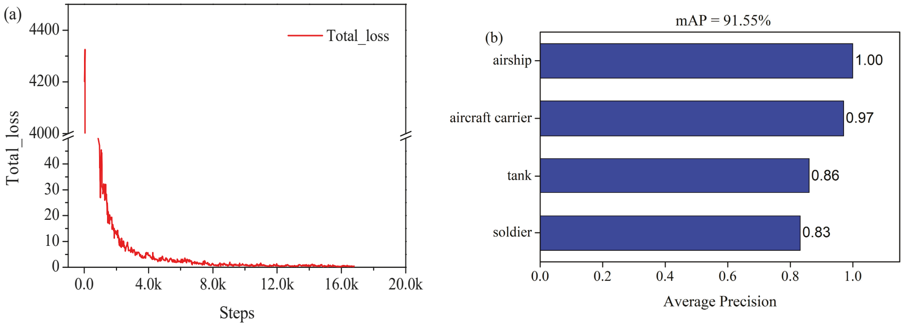


Figure 7. The total training loss and mean average precision. (**a**) the total loss; (**b**) the mAP.

Through training, our recognition precision of these four typical targets in different environments reached 91.55% (mAP@.5=91.55%), which highly met our application requirements. Moreover, this recognition process was just a demonstration. In practical applications, specific databases could be added according to the actual needs, and the recognition classes and precision could be increased through retraining (Figure 8). The recognition results of the model after our training are shown in Figure 8. As can be seen from Figure 8, our model could well identify four typical military targets. When the resolution of the input picture was 416 × 416, the model detection time was less than 25 ms.

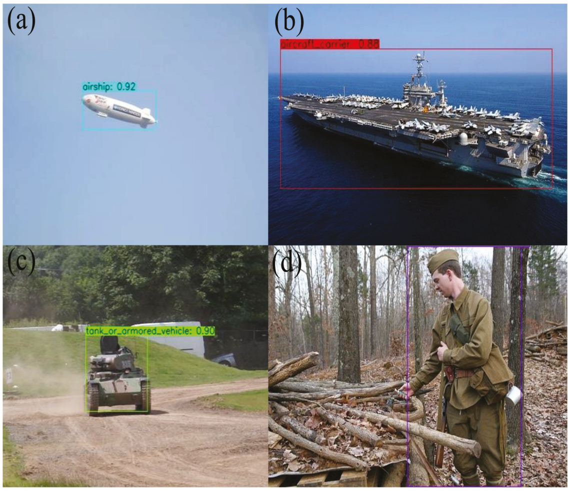


Figure 8. Object detection results. (**a**) airship; (**b**) aircraft carrier; (**c**) tank; (**d**) uniformed soldier.

4.2. Preliminary Camouflage Texture

We used the method described in Section 3.4 to design the initial camouflage texture. Firstly, we masked the target region detected by the YOLOv3, as shown in Figure 9b. Then, we input the masked image into the pre-train deepfillv1 model. The weights adopted by deepfillv1 were trained on the Place2 [40] data set. Places2 is a data set of scene images, containing 10 million pictures and more than 400 different types of scene environments, which could be used for visual cognitive tasks with the scenes and environments as application content, meeting our application requirements. We used the weight files from the literature [39] that were pre-trained on Place2 [40]. Finally, the complete image was obtained, which is called preliminary camouflage texture, as shown in Figure 9c. It could be seen from Figure 9 that the generated preliminary camouflage pattern had a texture consistent with the environment, which could be well integrated into the environment. When the input image resolution is 416 × 416, the preliminary camouflage texture generation process takes less than 0.2 s.

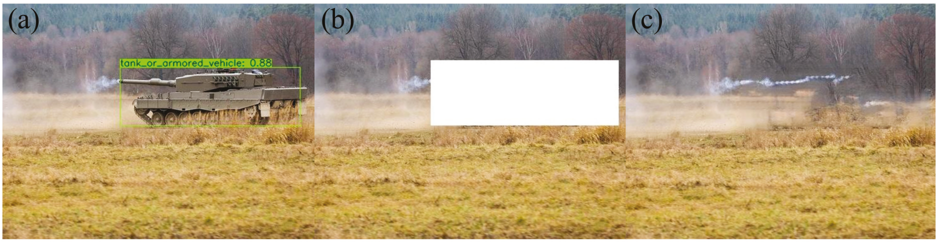


Figure 9. Preliminary camouflage texture design. (**a**) original image; (**b**) masked image; (**c**) completed image.

The more detailed experimental results are shown in Figure 10, where the first column shows the original images, with rectangular bounding boxes indicating the targets to be camouflaged, the second column shows the results of the preliminary camouflage texture design.

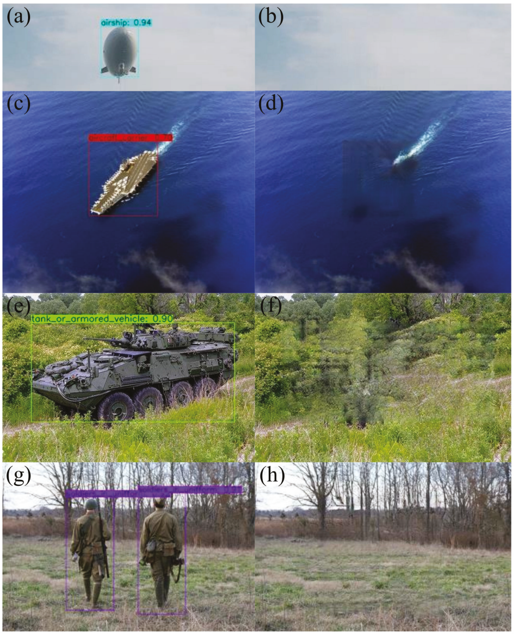


Figure 10. More detailed experimental results. (**a**) airship; (**b**) preliminary camouflage texture of the airship; (**c**) aircraft carrier; (**d**) preliminary camouflage texture of the aircraft carrier; (**e**) tank; (**f**) preliminary camouflage texture of the tank; (**g**) uniformed soldiers; (**h**) preliminary camouflage texture of the uniformed soldiers.

4.3. Standardization of Camouflage Texture

We standardized the initial camouflage texture using the method described in Section 3.5. As shown in Figure 11b, the camouflage texture we designed corresponded to the rectangular area where the target is located. Note that K is set to 5 when discussed later in this article. At the same time, we needed to design the camouflage texture for the camouflage target's forward view, backward view, left view, right view, and top view, respectively. In practice, the texture of the camouflage area needed to be mapped to the actual target surface. This step could be accomplished with 3D rendering software, such as Maya or OpenGL, which is not described in this article as it focuses on the design approach. In this article, we simply mapped the camouflage texture to the camouflage target surface through a mask to observe its camouflage effect, as shown in Figure 11c. The output camouflage textures in Figure 11c had an overall optical camouflage effect, where textures and semantics were consistent with the environment look very naturally.

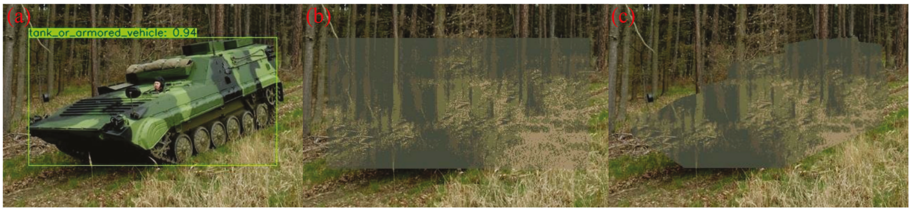


Figure 11. Adaptive digital camouflage texture. (**a**) Original image; (**b**) Rectangular texture area; (**c**) Camouflaged images using textures.

The more detailed experimental results are shown in Figure 12. The camouflage texture generation process in this paper is different from the traditional one. The traditional camouflage texture was obtained by random or non-random distribution using finite pattern templates or structural elements of texture. The camouflage texture generation in this paper is to use the method of image completion to generate the texture consistent with the current environment. The texture composition of this paper had no fixed structural elements. It might have been random for the natural environment like forest or regular for the artificial environment like the city, depending on the state of the current environment. The texture features come from a lot of training we did on the places2 [40] data set using the deepfillv1 [39] algorithm. The place2 [40] data set contains more than 400 different types of scene environments and 10 million images. Basically, it covers people's common scenes. The deepfillv1 [39] algorithm, after training on places2 [40], was able to generate a meaningful image consistent with the background texture of the incomplete image. As shown in Figure 12, camouflage textures are designed using this method on both natural and artificial backgrounds. In Figure 12, the first column shows the original images in natural and artificial environments, and the second column shows the camouflaged image corresponding to the first column. The color of the second column camouflage texture was not replaced by the standard color. The third column shows the camouflaged image with the camouflaged texture of the standard color. In Figure 12, the first row shows the camouflage texture in the natural environment using our method, the second row shows the camouflage texture in the artificial environment using our method. As can be seen from Figure 12, the camouflage texture designed using the method we provided has an excellent camouflage effect in both natural and artificial environments. The camouflage texture in the natural environment is irregular, and the camouflage texture in the artificial environment has a certain rule, which is consistent with the current environment. Comparing Figure 12e,f, it is found that the camouflage performance decreased after filling the camouflage texture with the most similar standard color. This is because the standard colors we choose are only 30 colors specified in the standard, which is a little different from the main colors of the current environment. This does not affect the effectiveness of the design method we provide. With the development of controllable color change technology, we may be able to choose far more than 30 colors in the future. At that time, the camouflage performance of the camouflage texture designed by this method would be further improved.

When the input image resolution was 416 × 416, the standardization of the camouflage texture took 0.1 s. All the tests were implemented on Python3.6, TensorFlow v1.6, CUDNN v7.1, CUDA v9.2, and run on hardware with CPU Intel Core I7-9700F (3.0 GHz) and GPU RTX 2080 Ti. Meanwhile, the whole camouflage texture generation process took less than 0.4 s. This time could be shortened significantly with the improvement of the image completion algorithms or the improvement of hardware performance. We firmly believe that real-time methods will be proposed in the near future. This shows that the method provided in this paper could quickly generate camouflage texture in real-time, which could be used in future combat equipment and personnel adaptive camouflage design.

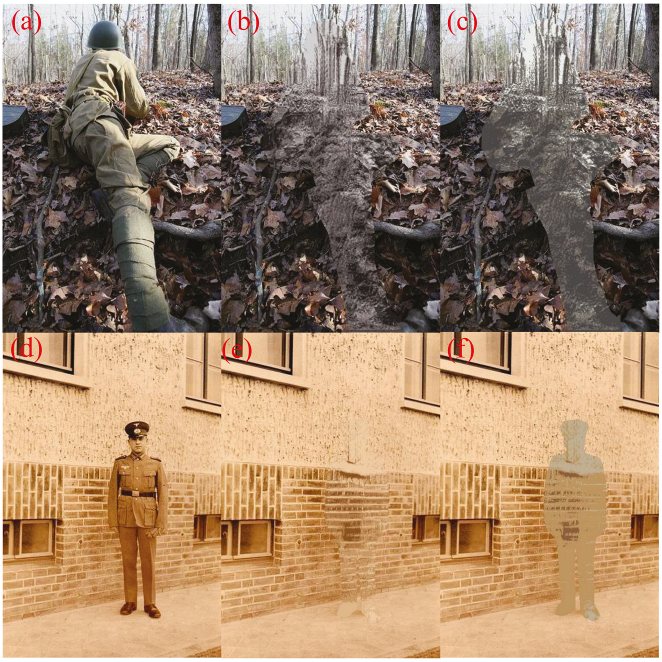


Figure 12. Camouflage textures in natural and artificial environments. (**a**) the original image in natural environment; (**b**) the camouflaged image corresponding to (**a**) whose color wasn't replaced by standard color; (**c**) the camouflaged image corresponding to (**a**) whose color was replaced by standard color; (**d**) the original image in artificial environment; (**e**) the camouflaged image corresponding to (**d**) whose color wasn't replaced by standard color; (**f**) the camouflaged image corresponding to (**d**) whose color was replaced by standard color.

5. Discussion

Visual saliency is the perceptual quality that makes an object, person, or pixel stand out relative to its neighbors, and thus captures our attention. Therefore, it is a reasonable and effective method to evaluate the camouflage performance of a camouflaged target by the saliency detection algorithm, which has been used in many literatures [5,46,47]. In this paper, the frequency-tuned salient region detection (FT) [48] algorithm, as a classic saliency detection algorithm, was used to quantitatively evaluate the performance of camouflage texture. The saliency map of the camouflaged target was obtained after FT algorithm detection. The higher the salience value was, the more conspicuous was the foreground target, and the weaker was the camouflage effect.

To verify the validity and effectiveness of our proposed design method, we compared the saliency map of the camouflage texture designed using the traditional design method in the literature [5] with the camouflage texture designed using the method we provided, as shown in Figure 13. There are five images in Figure 13, where (a) shows an original image with foreground targets highlighted with red rectangles, (b) shows the results of camouflaging the targets using the camouflage texture designed by the existing method [5], (c) shows the results of manually camouflaging the targets using the camouflage texture designed by the method provided by us, (d) shows the saliency map corresponding to the image (b), (e) shows the saliency map corresponding to the image (c). As we can clearly see that the camouflage texture designed with our method has better camouflage performance, since in Figure 13d, the target contour and the mosaic-like stripe of the camouflage texture can be clearly distinguished, while in Figure 13e, the camouflage target could hardly be distinguished. Note that all images above have the same resolution. This difference is due to the camouflage texture designed by the method in literature [5] being inconsistent with the background texture and semantics, while the texture designed by the method we provide is consistent with the background texture and semantics. This is because we are able to learn features from the surrounding environment using the deepfillv1 algorithm, whereas the existing method is a texture template based on empirical design.

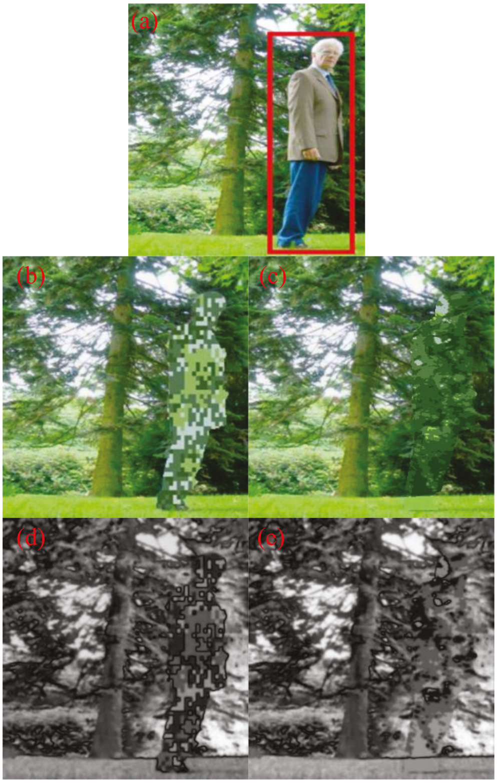


Figure 13. Comparison between our method and an existing method. (**a**) original image; (**b**) traditional camouflage texture; (**c**) adaptive camouflage texture; (**d**) the saliency map corresponding to image (**b**); (**e**) the saliency map corresponding to image (**c**).

As shown in Figure 14, in order to evaluate the camouflage performance of the camouflage texture designed by our method, we used the saliency algorithm FT to calculate the salience map of the images before and after camouflage. In Figure 14, the first column shows the original image, the second column shows the salience map corresponding to the first column, the third column shows the camouflaged images using a texture designed by our method, the fourth column shows the salience map corresponding to the third column. As we can see from Figure 14, the designed camouflage texture satisfies the color condition: (1) the main color has different brightness, and (2) the main color is not much different from the background color. At the same time, the camouflage texture and the background have texture and semantic consistency. Therefore, the camouflage texture designed with the method we provided could blend well into the background.

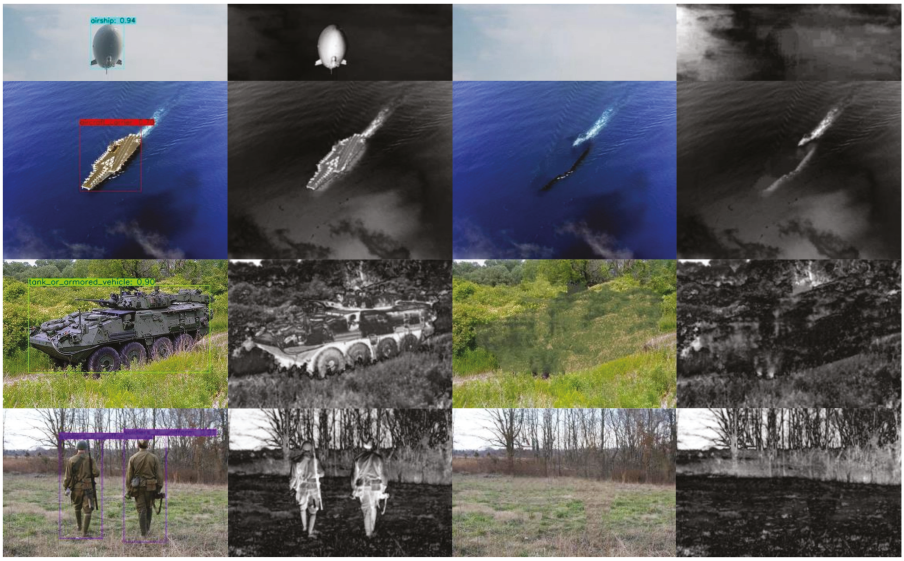


Figure 14. Evaluate the performance of the camouflage image using the salience map.

In contrast with the original image, the foreground targets are almost indistinguishable in the camouflaged image's salience map, suggesting that the camouflaged targets blend well into the background and are invisible to human eyes and visible light reconnaissance equipment. At the same time, we input the camouflaged image into the YOLOv3 model of the previous training for reidentification. The results show that the target in the camouflaged image cannot be recognized by the YOLOv3 model. The experimental results show that the adaptive digital camouflage with excellent camouflage performance could be designed quickly with our design method.

6. Conclusions

Adaptive optical camouflage technology is the inevitable direction of future optical camouflage technology development. As one of the key technologies of adaptive optical camouflage, the study of adaptive camouflage texture design has important theoretical and practical significance. In this paper, a fast-self-adaptive digital camouflage design method based on the neural network is proposed for the new generation of self-adaptive optical camouflage. First, we used the YOLOv3 algorithm to train the recognition model of four typical military targets. After adjusting the hyper-parameters, we got a model with good recognition probability, whose mean average precision (mAP) was 91.6%. Then, we used the deepfillv1 algorithm to do the preliminary camouflage texture design for the recognition area. Finally, the clustering algorithm was used to extract the main color of the camouflage target region, and the most similar color in the standard is used to standardize the color in the initial texture. The camouflage texture designed by our method was consistent with the texture and semantics of the real-time background. The whole texture generation process is very short, less than 0.4 s, which could meet the requirements of near-real-time camouflage design in the future. The saliency detection results showed that the camouflage texture generated by the proposed method had good camouflage performance in optical camouflage. At present, the method is effective for camouflage design in forest, grassland, desert, and other natural environments. But in artificial environment, such as urban environment, the effect of camouflage design is not very ideal. In addition, there are not many typical target images available and relevant datasets need to be further collected. In the future, on the one hand, we will further study how to improve the camouflage design performance of this method in an artificial environment. On the other hand, the implementation of adaptive camouflage systems will be further studied, such as the control system of adaptive camouflage. Nevertheless, this paper proposes and implements a new idea of adaptive camouflage texture design, which has important potential application value in future real-time optical camouflage.

Author Contributions: Conceptualization, H.X.; Data curation, H.X.; Formal analysis, H.X.; Investigation, Z.Q.; Methodology, H.X. Z.Q.; Project administration, M.L.; Supervision, M.L.; Visualization, Y.J., C.W. and R.Q.; Writing—original draft, H.X.; Writing—review & editing, Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Talas, L.; Baddeley, R.J.; Cuthill, I.C. Cultural evolution of military camouflage. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **2017**, *372*, 20160351. [CrossRef]
2. Merilaita, S.; Scott-Samuel, N.E.; Cuthill, I.C. How camouflage works. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **2017**, *372*, 20160341. [CrossRef] [PubMed]
3. King, A. The digital revolution: Camouflage in the twenty-first century. *Millenn. J. Int. Stud.* **2014**, *42*, 397–424. [CrossRef]
4. Chu, M.; Tian, S.H. An Extraction Method for Digital Camouflage Texture Based on Human Visual Perception and Isoperimetric Theory. In Proceedings of the 2nd International Conference on Image, Vision and Computing, Chengdu, China, 2–4 June 2017; pp. 158–162.
5. Xue, F.; Xu, S.; Luo, Y.T.; Jia, W. Design of digital camouflage by recursive overlapping of pattern templates. *Neurocomputing* **2016**, *172*, 262–270. [CrossRef]
6. Zylinski, S.; Darmaillacq, A.S.; Shashar, N. Visual interpolation for contour completion by the European cuttlefish (Sepia officinalis) and its use in dynamic camouflage. *Proc. R. Soc. B Biol. Sci.* **2012**, *279*, 2386–2390. [CrossRef]
7. Kelman, E.J.; Osorio, D.; Baddeley, R.J. A review of cuttlefish camouflage and object recognition and evidence for depth perception. *J. Exp. Biol.* **2008**, *211*, 1757–1763. [CrossRef]
8. Barbosa, A.; Allen, J.J.; Mäthger, L.M.; Hanlon, R.T. Cuttlefish use visual cues to determine arm postures for camouflage. *Proc. R. Soc. B Biol. Sci.* **2012**, *279*, 84–90. [CrossRef]
9. Allen, J.J.; Mäthger, L.M.; Barbosa, A.; Buresch, K.C.; Sogin, E.; Schwartz, J.; Chubb, C.; Hanlon, R.T. Cuttlefish dynamic camouflage: Responses to substrate choice and integration of multiple visual cues. *Proc. Biol. Sci.* **2010**, *277*, 1031–1039. [CrossRef]
10. Teyssier, J.; Saenko, S.V.; Van Der Marel, D.; Milinkovitch, M.C. Photonic crystals cause active colour change in chameleons. *Nat. Commun.* **2015**, *6*, 1–7. [CrossRef]
11. Vigneron, J.P.; Pasteels, J.M.; Windsor, D.M.; Vértesy, Z.; Rassart, M.; Seldrum, T.; Dumont, J.; Deparis, O.; Lousse, V.; Biro, L.P.; et al. Switchable reflector in the Panamanian tortoise beetle Charidotella egregia (Chrysomelidae: Cassidinae). *Phys. Rev. E Stat. Nonlin. Soft Matter. Phys.* **2007**, *76*, 031907. [CrossRef]
12. Zhao, Y.; Xie, Z.; Gu, H.; Zhu, C.; Gu, Z. Bio-inspired variable structural color materials. *Chem. Soc. Rev.* **2012**, *41*, 3297–3317. [CrossRef] [PubMed]
13. Morin, S.A.; Shepherd, R.F.; Kwok, S.W.; Stokes, A.A.; Nemiroski, A.; Whitesides, G.M. Camouflage and display for soft machines. *Science* **2012**, *337*, 828–832. [CrossRef] [PubMed]
14. Wang, G.; Chen, X.; Liu, S.; Wong, C.; Chu, S. Mechanical Chameleon through Dynamic Real Time-Plasmonic Tuning. *Acs Nano* **2016**, *10*, 1788–1794. [CrossRef] [PubMed]
15. Arsenault, A.C.; Míguez, H.; Kitaev, V.; Ozin, G.A.; Manners, I. Towards photonic ink (P-Ink): A polychrome, fast response metallopolymer gel photonic crystal device. *Macromol. Symp.* **2003**, *196*, 63–69. [CrossRef]
16. Arsenault, A.C.; Puzzo, D.P.; Manners, I.; Ozin, G.A. Photonic-crystal full-colour displays. *Nat. Photonics* **2007**, *1*, 468–472. [CrossRef]
17. Puzzo, D.P.; Arsenault, A.C.; Manners, I.; Ozin, G.A. Electroactive Inverse Opal: A Single Material for All Colors. *Angew. Chem. Int. Edit.* **2009**, *48*, 943–947. [CrossRef]

18. Kim, H.; Ge, J.; Kim, J.; Choi, S.E.; Lee, H.; Lee, H.; Park, W.; Yin, Y.; Kwon, S. Structural colour printing using a magnetically tunable and lithographically fixable photonic crystal. *Nat. Photonics* **2009**, *3*, 534–540. [CrossRef]
19. Yang, H.F.; Yin, J.P. An Adaptive Digital Camouflage Scheme Using Visual Perception and K-Mean Clustering. In Proceedings of the 3rd International Conference on Materials and Products Manufacturing Technology Guangzhou, Guangzhou, China, 25–26 September 2013; pp. 1091–1094.
20. Simonyan, K.; Zisserman, A. Very deep convolution networks for large-scale image recognition. *arXiv e-prints* **2014**, arXiv:1409.1556.
21. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
22. Liu, Y.; Tao, Z.; Zhang, J.; Hao, H.; Peng, Y.; Hou, J.; Jiang, T. Deep-Learning-Based Active Hyperspectral Imaging Classification Method Illuminated by the Supercontinuum Laser. *Appl. Sci. Basel* **2020**, *10*, 17. [CrossRef]
23. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
24. Murthy, C.B.; Hashmi, M.F.; Bokde, N.D.; Geem, Z.W. Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms-A Comprehensive Review. *Appl. Sci. Basel* **2020**, *10*, 46. [CrossRef]
25. Prappacher, N.; Bullmann, M.; Bohn, G.; Deinzer, F.; Linke, A. Defect Detection on Rolling Element Surface Scans Using Neural Image Segmentation. *Appl. Sci. Basel* **2020**, *10*, 13. [CrossRef]
26. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
27. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
28. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef] [PubMed]
29. Iizuka, S.; Simo-Serra, E.; Ishikawa, H. Globally and locally consistent image completion. *Acm T Graphic* **2017**, *36*, 1–14. [CrossRef]
30. Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; Huang, T.S. Free-Form Image Inpainting with Gated Convolution. *arXiv e-prints* **2019**, arXiv:1806.03589.
31. Zhao, D.; Guo, B.L.; Yan, Y.Y. Parallel Image Completion with Edge and Color Map. *Appl. Sci. Basel* **2019**, *9*, 29. [CrossRef]
32. Pezeshkian, N.; Neff, J.D. Adaptive electronic camouflage using texture synthesis. In Proceedings of the Conference on Unmanned Systems Technology XIV, Baltimore, MD, USA, 25–27 April 2012.
33. Inami, M.; Kawakami, N.; Tachi, S. Optical camouflage using retro-reflective projection technology. In Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, Tokyo, Japan, 7–10 October 2003; pp. 348–349.
34. Uema, Y.; Koizumi, N.; Chang, S.W.; Minamizawa, K.; Sugimoto, M.; Inami, M. Optical Camouflage III: Auto-Stereoscopic and Multiple-View Display System using Retro-Reflective Projection Technology. In Proceedings of the 19th IEEE Virtual Reality Conference, Costa Mesa, CA, USA, 4–8 March 2012; pp. 57–58.
35. Yu, C.; Li, Y.; Zhang, X.; Huang, X.; Malyarchuk, V.; Wang, S.; Shi, Y.; Gao, L.; Su, Y.; Zhang, Y.; et al. Adaptive optoelectronic camouflage systems with designs inspired by cephalopod skins. *Proc. Natl. Acad. Sci. USA* **2014**, *111*, 12998–13003. [CrossRef]
36. Zhang, Y.; Xue, S.Q.; Jiang, X.J.; Mu, J.Y.; Yi, Y. The Spatial Color Mixing Model of Digital Camouflage Pattern. *Def. Technol.* **2013**, *9*, 157–161. [CrossRef]
37. Jia, Q.; Xu, W.D.; Hu, J.H.; Liu, J.; Yang, X.; Zhu, L.Y. Design and evaluation of digital camouflage pattern by spot combination. *Multimed. Tools Appl.* **2020**, *5*, 18. [CrossRef]
38. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv e-prints* **2018**, arXiv:1804.02767.

39. Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; Huang, T.S. Generative Image Inpainting with Contextual Attention. *arXiv e-prints* **2018**, arXiv:1801.07892.
40. Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; Torralba, A. Places: A 10 Million Image Database for Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 1452–1464. [CrossRef] [PubMed]
41. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE-Computer-Society Conference on Computer Vision and Pattern Recognition Workshops, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
42. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 27–30 June 2016; pp. 779–788.
43. Joseph, R.; Ali, F. Yolo9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
44. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. *arXiv e-prints* **2017**, arXiv:1701.07875.
45. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved Training of Wasserstein GANs. *arXiv e-prints* **2017**, arXiv:1704.00028.
46. Feng, X.; Guoying, C.; Richang, H.; Jing, G. Camouflage texture evaluation using a saliency map. *Multimed. Syst.* **2015**, *21*, 169–175. [CrossRef]
47. Cheng, X.P.; Zhao, D.P.; Yu, Z.J.; Zhang, J.H.; Bian, J.T.; Yu, D.B. Effectiveness evaluation of infrared camouflage using image saliency. *Infrared. Phys. Technol.* **2018**, *95*, 213–221. [CrossRef]
48. Achanta, R.; Hemami, S.; Estrada, F.; Susstrunk, S. Frequency-tuned Salient Region Detection. In Proceedings of the IEEE-Computer-Society Conference on Computer Vision and Pattern Recognition Workshops, Miami, FL, USA, 20–25 June 2009; pp. 1597–1604.

applied sciences

MDPI

Article

Remote Sensing Scene Classification and Explanation Using RSSCNet and LIME

Sheng-Chieh Hung [1], Hui-Ching Wu [2] and Ming-Hseng Tseng [1,3,4,*]

[1] Master Program in Medical Informatics, Chung Shan Medical University, Taichung 402, Taiwan; s0859003@gm.csmu.edu.tw
[2] Department of Medical Sociology and Social Work, Chung Shan Medical University, Taichung 402, Taiwan; graciewu@csmu.edu.tw
[3] Department of Medical Informatics, Chung Shan Medical University, Taichung 402, Taiwan
[4] Information Technology Office, Chung Shan Medical University Hospital, Taichung 402, Taiwan
* Correspondence: mht@csmu.edu.tw; Tel.: +886-424-730-022 (ext. 12214)

Received: 23 July 2020; Accepted: 2 September 2020; Published: 4 September 2020

Abstract: Classification is needed in disaster investigation, traffic control, and land-use resource management. How to quickly and accurately classify such remote sensing imagery has become a popular research topic. However, the application of large, deep neural network models for the training of classifiers in the hope of obtaining good classification results is often very time-consuming. In this study, a new CNN (convolutional neutral networks) architecture, i.e., RSSCNet (remote sensing scene classification network), with high generalization capability was designed. Moreover, a two-stage cyclical learning rate policy and the no-freezing transfer learning method were developed to speed up model training and enhance accuracy. In addition, the manifold learning t-SNE (t-distributed stochastic neighbor embedding) algorithm was used to verify the effectiveness of the proposed model, and the LIME (local interpretable model, agnostic explanation) algorithm was applied to improve the results in cases where the model made wrong predictions. Comparing the results of three publicly available datasets in this study with those obtained in previous studies, the experimental results show that the model and method proposed in this paper can achieve better scene classification more quickly and more efficiently.

Keywords: neural network; deep learning; cyclical learning rate; remote sensing; scene classification

1. Introduction

With the gradual advancement of technology today, smart mobile devices and aerial cameras are beginning to appear on the market. As the performance of the hardware improves, aerial photography technology is constantly improving along with it, and rapid breakthroughs in imaging technology have made it possible to acquire imagery quickly. There are more types of imagery than ever before and the imagery is also clearer than was possible previously. Remote sensing images can be used in many technical aspects of scene classification, such as land-cover detection, urban planning, disaster relief, traffic control, etc. Therefore, how to classify large amounts of remote sensing image data covering land areas is an important research topic [1–8].

In the past, when using image data for classification research, the primary focus was on how to effectively perform the task of feature extraction [9,10]. The deep-learning methods used today make use of different convolutional neural network models to automatically perform feature recognition, extract the required image details, and then train the classification model to recognize the scene. The popularization of graphics cards in recent years has enabled the amount of time spent on deep learning in neural network model training to be reduced. It has also enabled the rapid development of deep-learning research and studies related to the classification of high-resolution remote sensing images [11–13].

Related parameters used in deep-learning models include the training methods, network architecture, optimizer design, hardware operation, etc. Adjusting the model training hyper-parameters is a very important aspect of the model design. Smith [14] proposed a new learning-rate method; instead of a fixed value, a cyclical learning policy was set for the model being trained. The results showed that this could effectively reduce the number of training iterations and improve the accuracy of the classification. Smith and Topin [15] then proposed a new super-convergence one-cycle cyclical learning policy and suggested the usage of a large learning rate for model training, which, according to them, can improve the model's generalization capability. More recently, Leclerc and Madry [16] explored the impact of different learning rates on deep learning and found that the low and high values of cyclical learning rates [14,15] concur with their two regimes.

Training is the process of learning and adjusting the parameters of a model. During training, iteration methods such as the gradient descent learning algorithm are commonly used. Early stopping [17] is a method often used during training to prevent overfitting. This method calculates the accuracy of the test dataset during model training. When the accuracy of the test data no longer improves, the training stops and is terminated early. This not only helps to prevent overfitting of the training model but also improves the model's generalization ability.

This study aimed to produce an improved model with enhanced detection ability and a small number of training iterations. A two-stage circular learning method for training was, therefore, proposed. First of all, the image features were obtained by transfer learning; the classification framework designed in this paper was then used for training. The two-stage circular learning rate was used to reduce the number of iterations and, finally, the best model weights were obtained using the early stopping method. The main contributions of this paper are as follows:

- In order to reduce model overfitting and to obtain models with a high generalization capability, we recommend a new CNN (convolutional neutral networks) architecture, i.e., RSSCNet (remote sensing scene classification network), integrated with the simultaneous use of a two-stage cyclical learning-rate training policy and the no-freezing transfer learning technology that requires only a small number of iterations. In this way, an excellent level of accuracy can be obtained.
- By using the LIME (local interpretable model, agnostic explanation) super-pixel explanation, the root causes of model classification errors were made clearer and a further understanding was obtained. After the image correction preprocessing on the misclassified cases in the WHU-RS19 [18] dataset, this correction procedure was found to improve the overall classification accuracy. We hope that readers can better understand the reasoning mechanism of AI models for remote sensing scene classification.

The remainder of this paper is organized as follows: the dataset is introduced in the second section. In the third section, the steps of the developed research method are explained in detail. The data results and results from other studies are discussed in the fourth section, along with an analysis of why improved results were obtained using the proposed method. The fifth section is the conclusion.

2. Datasets

In this study, three publicly available remote sensing image data sets—the UC Merced land-use dataset [19], RSSCN7 [20], and WHU-RS19 [18]—were used for testing the performance of the proposed method.

2.1. UC Merced Land-Use Dataset

The pixel resolution of the UC Merced land-use dataset is 1 ft (=0.3048 m), and the dataset contains a total of 2100 images. The images are composed of 21 different land-use types; each class of each image has 100 RGB color images. The land-use types include agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, and tennis court. All the images contain different textures and colors, as shown in Figure 1. The UC Merced land-use dataset images were converted into 224 × 224 pixel size for transfer learning.

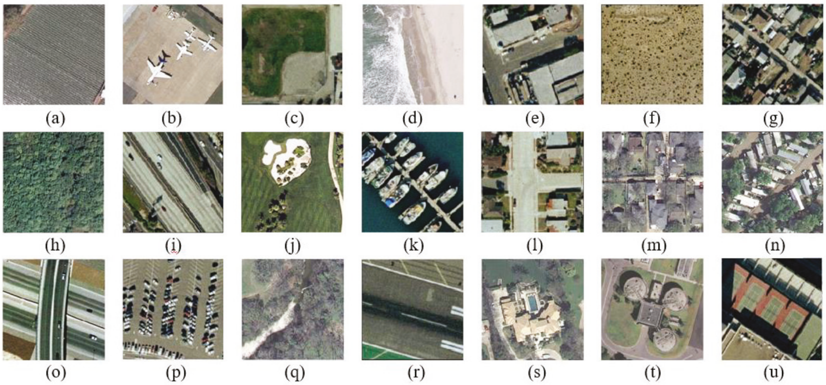


Figure 1. Example images of UC Merced dataset: (**a**) agriculture; (**b**) airplane; (**c**) baseball diamond; (**d**) beach; (**e**) buildings; (**f**) chaparral; (**g**) dense residential; (**h**) forest; (**i**) freeway; (**j**) golf course; (**k**) harbor; (**l**) intersection; (**m**) medium residential; (**n**) mobile home park; (**o**) overpass; (**p**) parking lot; (**q**) river; (**r**) runway; (**s**) sparse residential; (**t**) storage tanks; (**u**) tennis court.

2.2. RSSCN7 Dataset

The RSSCN7 dataset is a public dataset released by Wuhan University in 2015. There are seven different scene categories, including grass, field, industry, river, lake, forest, residential, and parking lot. The entire dataset includes a total of 2800 images. Each scene category of the dataset contains 400 images and corresponds to one of four different sampling ratios (1:700, 1:1300, 1:2600, and 1:5200); there are 100 images corresponding to each of these ratios. In the original dataset, all of the images have a size of 400 × 400 pixels. The data were acquired in different seasons and under various weather conditions. In the case of sampling differences in different proportions, classification of this dataset is a greater challenge. The different image categories are shown in Figure 2. The RSSCN7 dataset images were converted into 224 × 224 pixel size for transfer learning.

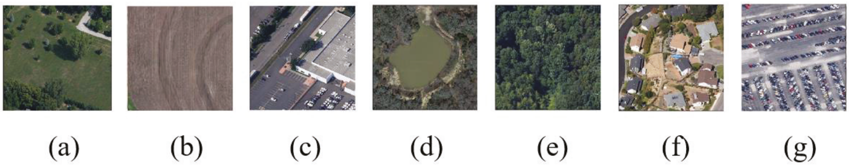


Figure 2. Example images of RSSCN7 dataset: (**a**) grass; (**b**) field; (**c**) industry; (**d**) river lake; (**e**) forest; (**f**) resident; (**g**) parking.

2.3. WHU-RS19 Dataset

The WHU-RS19 dataset was extracted from Google Earth satellite imagery. The spatial resolution of these satellite images is up to 0.5 m, and the spectral bands are red, green, and blue. There are 19 scene categories, including airport, beach, bridge, commercial, desert, farmland, football field, forest, industrial, meadow, mountain, park, parking, pond, port, railway station, residential, river, and viaduct. There are about 50 images corresponding to each category, and the entire dataset contains a total of 1005 images. The original image size is 600 × 600 pixels. Because the resolution, scale, direction, and brightness of the imagery vary greatly, processing this dataset is somewhat challenging. These data are also widely used in evaluating various scene classification methods. Figure 3 shows some samples from the dataset. The WHU-RS19 dataset images were converted into 224 × 224 pixel size for transfer learning.

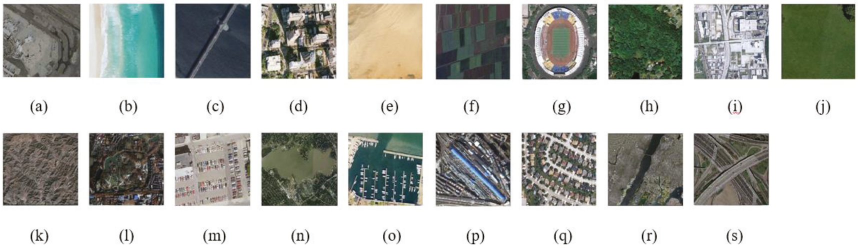


Figure 3. Example images of WHU-RS19 dataset: (**a**) airport; (**b**) beach; (**c**) bridge; (**d**) commercial; (**e**) desert; (**f**) farmland; (**g**) football field; (**h**) forest; (**i**) industrial; (**j**) meadow; (**k**) mountain; (**l**) park; (**m**) parking; (**n**) pond; (**o**) port; (**p**) railway station; (**q**) residential; (**r**) river; (**s**) viaduct.

3. Method

In this section, the model training method used in the experiments is introduced along with the convolutional neural network model used in this study and the two-stage cyclical learning-rate numerical design method used for the training.

3.1. Convolutional Neural Network Model

To start, the VGG [21] neural network trained by ImageNet was used as the model for image feature extraction. This method adjusts the structure of the neural network used for certain trained network models by using transfer learning to perform other image training tasks. In the experiments carried out in this study, the structure of the original neural network layer was adjusted by freezing the weights in one or more layers of the original model, minus the time to retrain the deep model. The original model was used for feature extraction, and the newly embedded model layer was trained for use in classification. It was, therefore, only necessary to update and modify the weights of the newly added network layer during training; the frozen layer weights that were transferred from the learning model could be kept, as shown in Figure 4 [22].

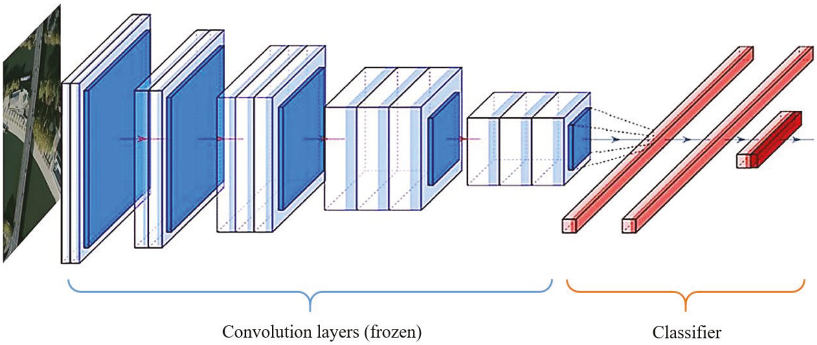


Figure 4. Convolutional neural network model.

After removing the top-level fully connected layers of the pre-training model, a new classification network was added. In our model, we chose an exponential linear unit (ELU) [23] as our activation function because it can reduce the vanishing gradient problem by identifying positive values. The ELU has negative values; hence, it allows the mean unit to approach zero for a deep neural network model and obtain a faster convergence than the rectified linear unit (ReLU). Regularization was also added as a tool to reduce overfitting in the network. Regularization can be considered as a penalty term in the loss function. The so-called "penalty" refers to restrictions that are applied to some parameters in the loss function to prevent overfitting. Moreover, we added a batch normalization layer [24] after the convolution layer in our model, which can act as a regularizer to decrease overfitting. Batch normalization can use a larger learning rate in model training to achieve faster convergence benefits.

Considering the balance between the network capacity and the test accuracy and discussing the influence of different regularization and optimization strategies, we finally designed this new deep learning network architecture with a high generalization capability. The proposed CNN architecture is called RSSCNet (Figure 5). In RSSCNet, the depth of the weight layers is 17, and the mathematical formulation is written as follows:

$$\left\{ \begin{array}{c} X = f^{(16)}\Big(f^{(15)}\Big(\cdots\Big(f^{(2)}\Big(f^{(1)}(\boldsymbol{x},\ \boldsymbol{w},\ \boldsymbol{b})\Big)\Big)\Big)\Big) \\ Y = Softmax(X) \end{array} \right\}, \tag{1}$$

where $\boldsymbol{x}$ is the input data of each image, $\boldsymbol{w}$ is the weight matrix, $\boldsymbol{b}$ is the bias, X is the representative feature of $\boldsymbol{x}$, and Y is the output probability. The RSSCNet architecture includes 15 convolutional layers, five max pooling layers, one global average pooling layer, two batch normalization layers [24], one dropout layer, and two fully connected layers with a softmax classification. Note that the activation function of the last two convolutional layers uses an ELU [23], while the other weight layers use the ReLU activation function. The convolution filter size is 3 × 3. The dropout rate is 0.5. The regulations of L1 and L2 are 0.01 and 0.02, respectively.

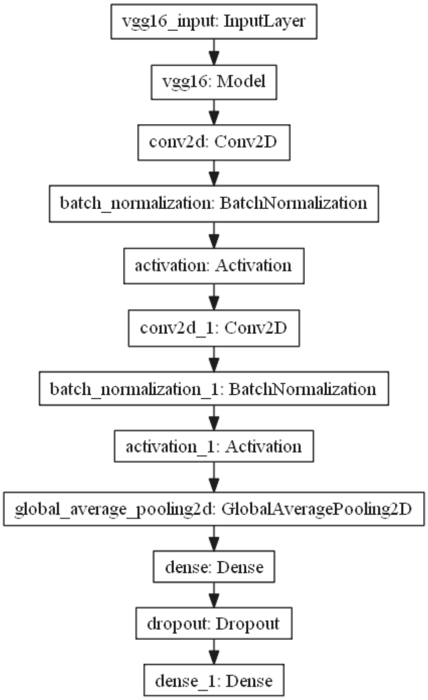


Figure 5. CNN classifier network architecture of our RSSCNet model.

3.2. Image Data Augmentation

When training deep-learning models, large amounts of data are needed for training, and it is necessary to try to avoid overfitting during the training. Proper use of regularization strategies related to deep learning, such as L1/L2 regularization, dropout, batch normalization, early stopping, and data augmentation, is needed. Among these strategies, data augmentation is regarded as an effective method for training a generally applicable model using limited training data [25]. In data augmentation, after the image is rotated, resized, scaled, and flipped, or has its brightness or color temperature changed, the original image in the dataset is changed to create more images that will allow the model to continue learning. In order to make up for the lack of data, in this study, an augmented training method was included in the training. After using horizontal and vertical flipping processing, the training image was increased by small-scale translation and scaling in order to enhance the generalization ability of the model.

3.3. Cyclical Learning Rate

The learning-rate method proposed by Smith [9] sets cyclical learning rates for the model instead of a fixed value and uses this to train the model. Results show that this can improve the accuracy of the classification and reduce the need for trivial adjustments. During training, the number of iterations used is usually reduced, and there are three different cyclical ways in which the learning-rate loop method can learn: “triangular”, “triangular2”, and “exp_range”. In our research, a two-stage cyclical learning-rate method was used to train the model. In the first stage, the “triangular” method was used to quickly find the best solution in the model; using this method, it was possible to avoid falling into a local solution when the learning rate was large. At the second stage, using the traingular2 method, the learning-rate cycle was gradually reduced to confine the model results until, finally, the solution stayed at a fixed position with no large swings (see Figure 6).

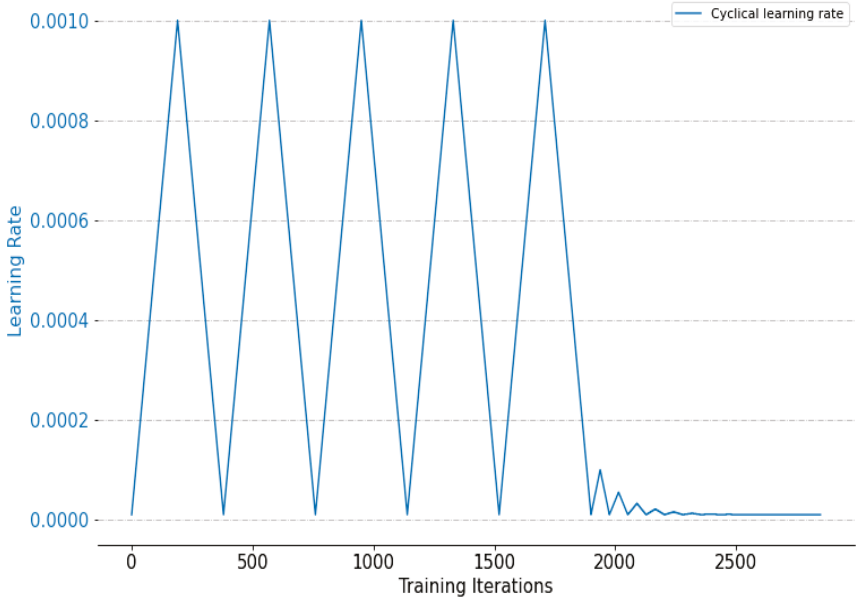


Figure 6. Cyclical learning rate during training.

The proposed two-stage cyclical learning rate method is calculated as shown in Equation (2).

$$\left\{\begin{array}{c} z = \left|1 + \frac{i}{\Delta} - 2 * \left[1 + \frac{i}{2\Delta}\right]\right| \\ D = \left\{\begin{array}{c} 1,\ for\ stage\ 1 \\ \left(\frac{lr_{min}}{lr_{max}}\right)^{\left(\frac{[\frac{i}{2\Delta}]}{[\frac{i_{max}}{2\Delta}]}\right)},\ for\ stage\ 2 \end{array}\right\} \\ lr = lr_{min} + (D * lr_{max} - lr_{min}) * max(0, (1 - z)) \end{array}\right\}, \tag{2}$$

where z is a dummy variable, i_{max} is the total number of training epochs i, Δ is the step size that is equal to the half cycle length, D is the damping factor, lr is the cyclical learning rate, lr_{min} is the minimum learning rate, and lr_{max} is the maximum learning rate.

3.4. t-Distributed Stochastic Neighbor Embedding (t-SNE) Analysis Method

The t-SNE analysis method is a nonlinear dimensionality reduction algorithm used for exploring high-dimensional data. Laurens van der Maaten and Geoffrey Hinton [26] proposed a new technique for visualizing similarity data in 2008. This technique can not only retain the partial structure of the data but can also display clusters of multiple scales at the same time. The t-SNE algorithm can project data into two-dimensional or three-dimensional space and uses good visualization to verify the effectiveness of the dataset or algorithm. The t-SNE method was used in various fields as a visualization method to evaluate the quality of classification [27,28]. It uses conditional probability and a Gaussian distribution to define the similarity between sample points in high and low dimensions and uses KL (Kullback–Leibler) divergence to measure the similarity between the sum of two conditional probability distributions; it also uses it a value function to decrease complexity by using the gradient method. The t-distribution is used to define the probability distribution at low dimensions to alleviate the congestion caused by dimensional disasters.

3.5. LIME Model Explanation Kit

Although a deep learning model can obtain quite good classification results, it is difficult to understand how the classification results are derived because of its black-box characteristics. How to interpret the reasoning mechanism of the deep-learning model has become an important topic of research. In recent years, among the deep-learning methods, LIME is a new evaluation method for the interpretability of the model [29], i.e., whether it is possible to understand the importance of the deep-learning model for the interpretability of the image in the subsequent classification and prediction. The problem with model interpretability is that it is difficult to define the decision boundary of the model in a way that humans can understand. As shown in Figure 7, LIME is a Python library that attempts to generate some local feature-circle super-pixels. This can be used to explain the principle on which the model is based, which is usually difficult to describe, and to help with understanding whether the basis on which the model applies its decisions is appropriate or not. Figure 7a shows that the most interesting super-pixel of the RSSCNet model contains an airplane; hence, it can be correctly classified by the RSSCNet model. Figure 7b depicts that there is no storage tank in the super-pixel unlike the RSSCNet model, thereby causing the RSSCNet model to make a misclassification.

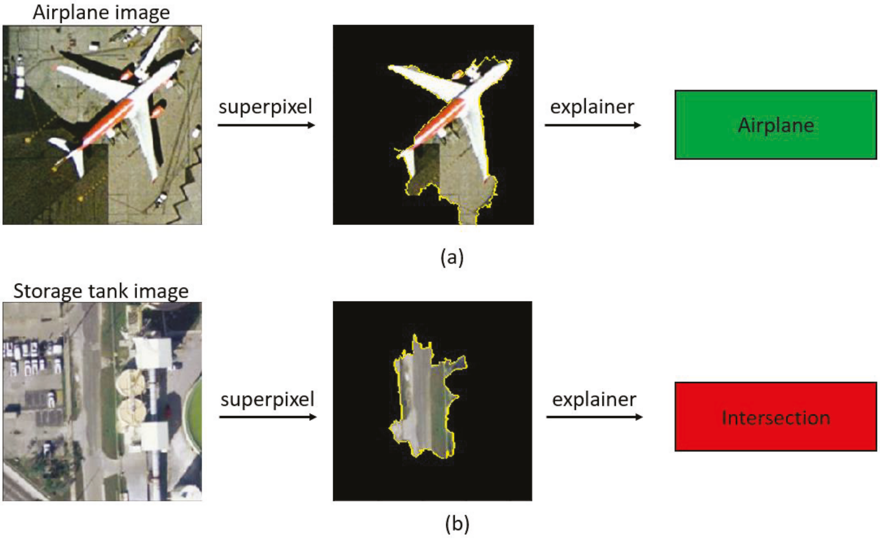


Figure 7. Image explanation using LIME (local interpretable model, agnostic explanation): (**a**) example of correct classification; (**b**) misclassified example.

4. Results and Analysis

4.1. Experimental Set-Ups

4.1.1. Implementation Details

In this study, the tensorflow2.0 suite within Python was used as the platform for the experiment. The hardware and system configuration included a Windows 10 version 1703 system. An NVIDIA GTX 1080 TI graphics card was used; the computing core was an i7-6700 3.40 GHz 8-core central processing unit (CPU) with 32 GB memory. Different pre-trained model parameter settings were used, and the results of using these were compared. Attempts were made to adjust the settings for training methods with different stages. The size of the batches in the experiment was set to a uniform size of 64, which was more in line with the memory capacity of the graphics card; the image length and width were set to 224 pixels in all cases. This study designed the training set size based on earlier studies. For the UC Merced land-use dataset, two training configurations—80% training and 50% training—were used. For the RSSCN7 dataset, 50% training and 20% training were used, and, for the WHU-RS19 dataset, the two modes used were 60% training and 40% training Two modes and 10 repeated random training cycles were used to verify and evaluate the experimental results.

4.1.2. Evaluation Methods

In this experiment, the confusion matrix and the overall accuracy were used to evaluate the classification performance, and the results were compared with those obtained using other, recently developed methods. The confusion matrix can be applied to the performance analysis of two-class or multi-class classification. After the model made its predictions, each class was assigned to one of a group of tables so that the data could be displayed and so that the detailed classification results for each category after the predictions were made could be seen. To evaluate the accuracy of the classification results, the overall accuracy was used. The accuracy ranged from 0 to 1, where a closer number to 1 denotes better classification performance. The total number of images that were correctly classified was divided by the number of test images.

In addition to the confusion matrix, the kappa coefficient is also often used to analyze the difference in the classification results for indicators of the multi-category classification quality. This coefficient is a method used in statistics to evaluate consistency. It calculates the index of the overall consistency and the classification consistency. The value range is [−1, 1]. A higher coefficient value denotes higher accuracy of the classification achieved by the model. The kappa coefficient (K) calculation formula with a higher degree is expressed as follows:

$$K = \frac{(P_0 - P_c)}{(1 - P_c)}. \tag{3}$$

4.2. Results and Analysis

4.2.1. Analysis of Experimental Parameters

In this study, different pre-trained models were tested for the evaluation of the classification results. This was done so that the best feature extraction method for the appropriate pre-trained model could be found. Once it was found, the weight layers in the pre-trained model were adjusted. It was considered whether the pre-trained model weights would affect the results of the transfer learning in order to find the best training-layer training plan for the model parameter configuration that would be used in subsequent experiments in this study.

1. Different pre-trained CNN models

Different pre-trained models have different degrees of influence on image feature extraction. In this experiment, the WHU-RS19 dataset was used to embed four different common pre-trained models—VGG16 [21], VGG19 [21], ResNet50 [30], and InceptionV3 [31]—into the classification model. A classification performance test was carried out to help decide which of the four models was the most suitable for use as the pre-trained model. The training used an Adam optimizer; the batch size was 64, and the number of iterations was 150. The results of the training carried out using the four different models are shown for comparison in Figure 8. The results show that the pre-trained VGG16 model had the best overall accuracy; thus, in subsequent testing, this was used as the image feature extraction model.

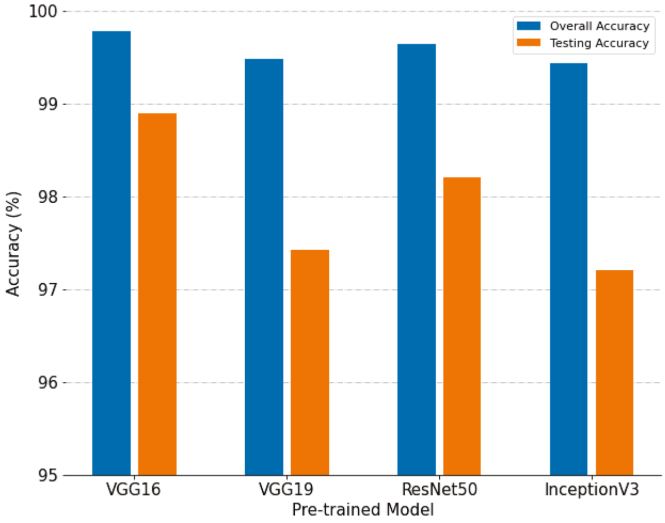


Figure 8. Comparison of accuracy using different pre-trained models.

2. Different numbers of fine-tuning layers during training

After choosing to use the VGG16 pre-trained model, this study further explored whether, by freezing some of the network layers in the pre-trained model, the model could be made to have a better generalization performance. This experiment was also conducted using the WHU-RS19 dataset, and the results are shown in Figure 9. Based on the four blocks contained in the VGG16 model, two, four, seven, 10, and 13 layers were frozen. The results show that, when no layers were frozen, all the layers of the pre-trained model were retrained and fine-tuned, which means that, although this method requires more resources and a longer training time, it can produce a better overall accuracy. This shows that, in the classification of remote sensing images, the feature image that is required can be obtained by further training.

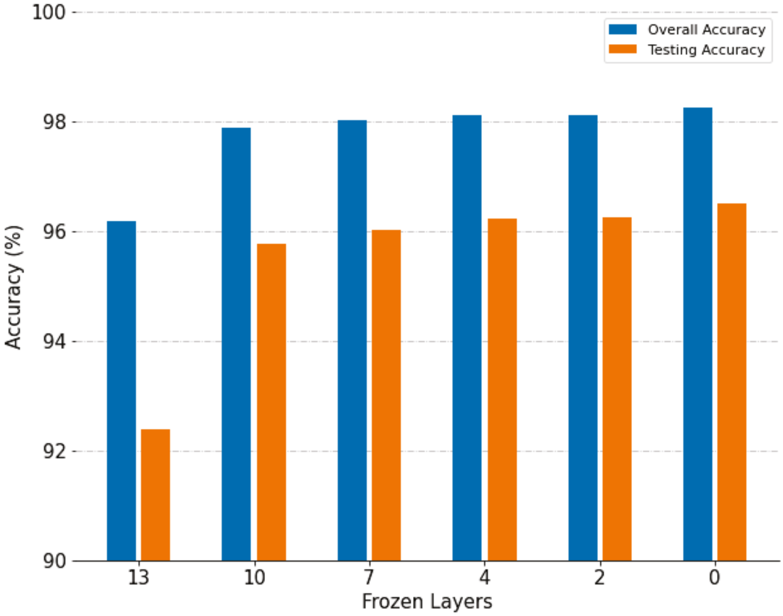


Figure 9. Comparison of accuracy using different fine-tuning layers.

For the different fine-tuning layers discussed, the main consideration was whether to retrain the weights in the pre-trained model. Retraining the entire network inevitably takes a lot of time. However, in the process of feature extraction, in addition to training, we believe that it is necessary to focus on the training of the classifier and, hence, in this study, we aimed to strengthen the model's ability to classify features by using a two-stage training method. In Figure 10, the WHU-RS19 dataset is used as the comparison dataset for the proposed method.

A two-stage training method (shown as "2-stage" in Figure 10) in our research indicates that two different optimizers were used and separated into two parts for the two-stage training. In the first stage, the SGD (stochastic gradient decent) optimizer was used to carry out 100 training iterations to train the entire neural network model. In this stage, the pre-trained model and classifier model could be adjusted at the same time, thus strengthening the features of the capture model and learning the classification ability. In the second stage of the training, the model weights with the best accuracy learned in the first stage were loaded, and the Adam optimizer was used to carry out the next 50 training iterations. We also compared the result with only using the Adam optimizer training for 150 iterations (shown as "1-stage" in Figure 10). As can be seen from Figure 10, the test accuracy obtained using the two-stage training (97.76%) was better than that obtained using the one-stage training (96.33%).

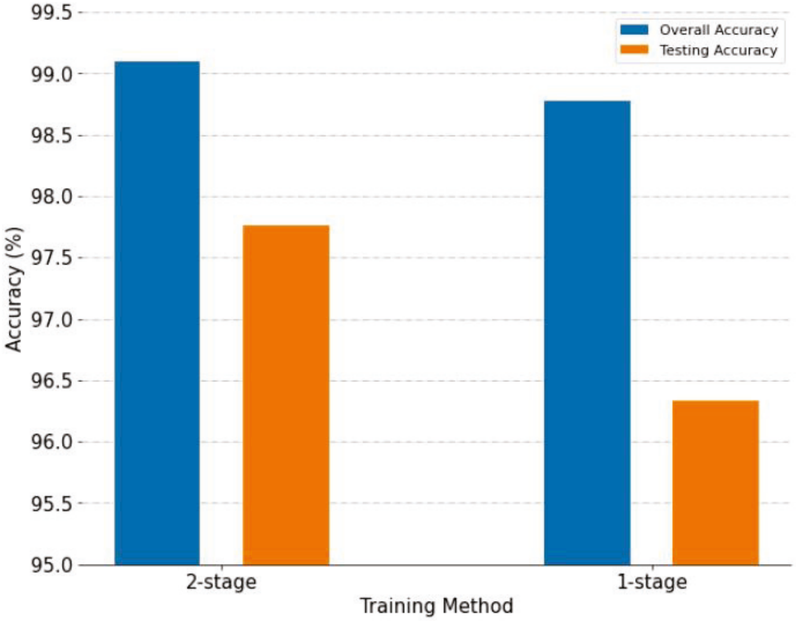


Figure 10. Comparison of overall accuracy with and without two-stage training.

3. Different classification architectures

For a performance comparison, we compare the RSSCNet architecture proposed herein with the other architectures in the literature, such as VGG-16-Net [21] and the GSB + LOB model [32]. Figure 11 showed the ranking results of the test accuracy of each model (i.e., RSSCNet: 0.978, VGG-16-Net: 0.973, and GSB + LOB model: 0.97), which indicated that the proposed RSSCNet is the best classification model.

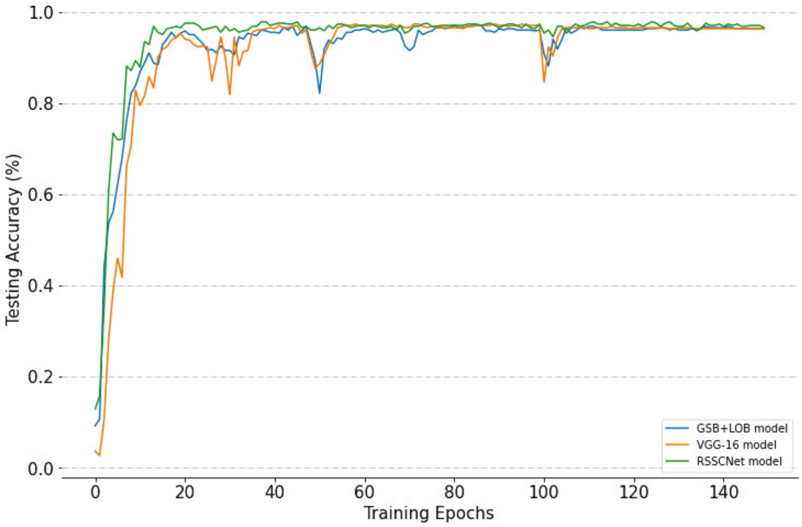


Figure 11. Comparison of testing accuracy using different classification architectures.

4. Different cyclical learning-rate methods

In this section, we compare the performances of the two-stage cyclical learning rate method (2-stage CLR, Figure 11) and Smith and Topin's (2019) one-cycle cyclical learning rate method (1-cycle CLR). The results of Figure 12 showed that the two-stage cyclical learning rate method achieved the highest test accuracy of 98.0% at epoch = 134. On the contrary, the one-cycle cyclical learning rate method only achieved the highest test accuracy of 97.0% at epoch = 27. Although the latter could provide quicker access to the best test accuracy for training, the former could achieve a better test accuracy; hence, it was used in subsequent experimental results for model training and performance testing.

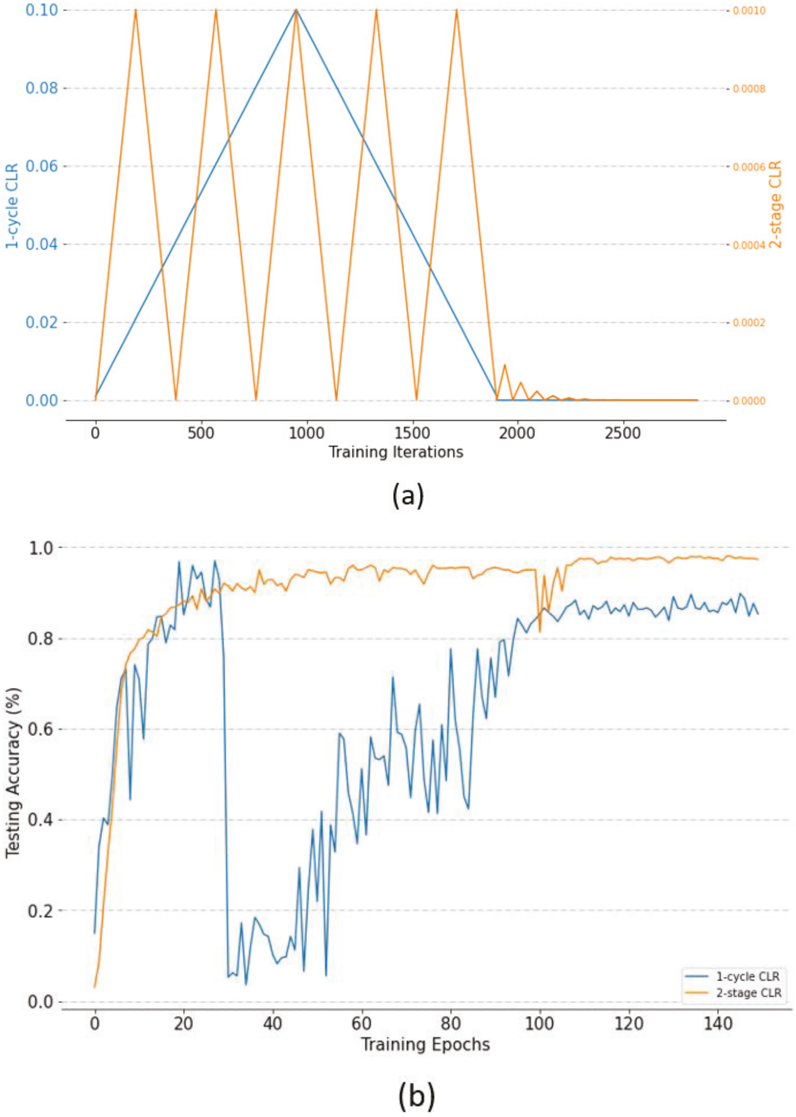


Figure 12. Comparison of testing accuracy using different cyclical learning rate (CLR) methods: (**a**) cyclical learning-rate policy; (**b**) testing accuracy.

4.2.2. Experimental Results

1. Classification of UC Merced land-use dataset

In this section, the classification results for the UC Merced land-use dataset are discussed. The t-SNE analysis method was used for the classification. This method is a non-linear dimensionality-reduction algorithm used for exploring high-dimensional data. It can map multi-dimensional data to two or more dimensions using technology suitable for visual presentation. In this study, extracting the features of the deep layers for analysis helped with the understanding of the differences between the features obtained by the model before and after training. In Figure 13a, which shows the results before training, the features extracted by the model show little correlation; however, after training, as shown in Figure 13b, the features are highly clustered, thus showing that these models do help to improve the classification performance.

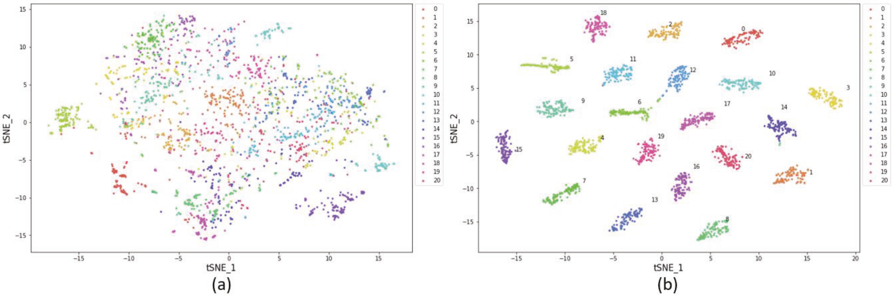


Figure 13. Visual analysis on UC Merced dataset using t-distributed stochastic neighbor embedding (t-SNE): (**a**) before training; (**b**) after training.

Figure 14 shows the VGG16 model supplemented by the model classification matrix proposed in this study using the best classification weights obtained from the early training termination strategy. The resulting confusion matrix is also shown; the training rate was 80%. The matrix contains the individual classification results for 21 categories. The kappa coefficients of the UC Merced dataset were 0.9985 and 0.9895 at 80% and 50% training, respectively.

In Table 1, the classification results found in this study are compared with those obtained using other classification methods. It can be seen that, by using the two-stage cyclical learning-rate training method, this study achieved the best overall accuracy out of the results shown.

Table 1. Comparison of the overall accuracy and standard deviations using 80% and 50% training ratios on UC Merced dataset.

Method	80% Training Ratio	50% Training Ratio
GoogLeNet [33]	94.31 ± 0.89	92.70 ± 0.60
CaffNet [33]	95.02 ± 0.81	93.98 ± 0.67
VGG-16 [33]	95.21 ± 1.20	94.14 ± 0.69
salM^{3}LBP-CLM [34]	95.75 ± 0.80	94.21 ± 0.75
CNN-R+VLAD with SVM [35]	95.85	NA
TEX-Net-LF [36]	96.62 ± 0.49	95.89 ± 0.37
VGG19+Hybrid-KCRC [37]	96.33	NA
Two-Stream Fusion [38]	98.02 ± 1.03	96.97 ± 0.75
CTFCNN [39]	98.44 ± 0.58	NA
GCFs + LOFs [32]	99.00 ± 0.35	97.37 ± 0.44
Inception-v3-CapsNet [40]	99.05 ± 0.24	97.59 ± 0.16
MVFLN+VGG-VD16 [41]	99.52 ± 0.17	NA
RSSCNet (this paper)	99.81 ± 0.06	98.76 ± 0.19

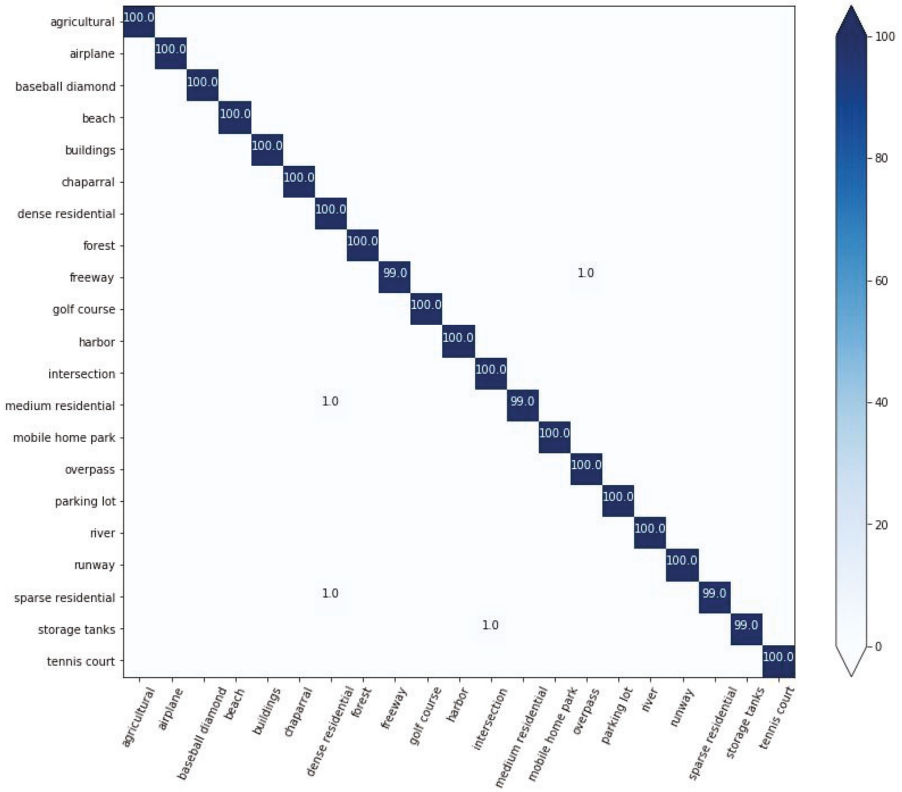


Figure 14. Classification confusion matrix of our method on UC Merced dataset.

2. Classification of RSSCN7 dataset

The t-SNE analysis method was used to extract the deep features of the proposed model and to analyze it. In this section, the classification results obtained by applying this model to the RSSCN7 dataset are discussed. As shown in Figure 15b, after training, the features became highly clustered, which shows that the model proposed by this research helps to improve the scene classification.

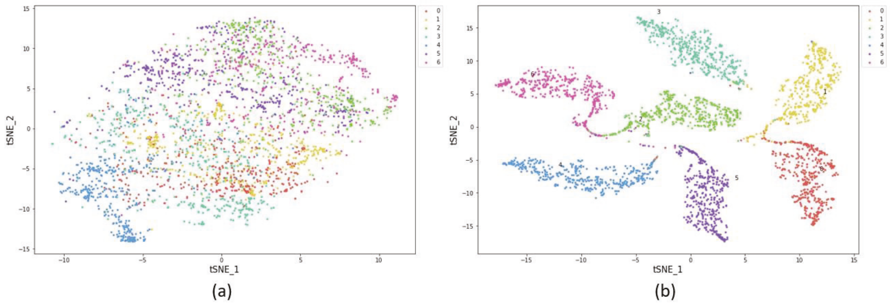


Figure 15. Visual analysis on RSSCN7 dataset using t-SNE: (**a**) before training; (**b**) after training.

Figure 16 shows the overall accuracy confusion matrix extracted by VGG16 supplemented by the classification method proposed in this study and using the optimal classification weights in the training early termination strategy. The training rate used was 50%. The matrix contains the individual classification results for seven categories. Among these, agricultural land and grassland are most likely to be confused. This is perhaps because the two categories have similar characteristics—both containing a large proportion of green ground, which easily leads to errors. The kappa coefficients of the RSSCN7 dataset were 0.9737 and 0.9329 at 50% and 20% training, respectively.

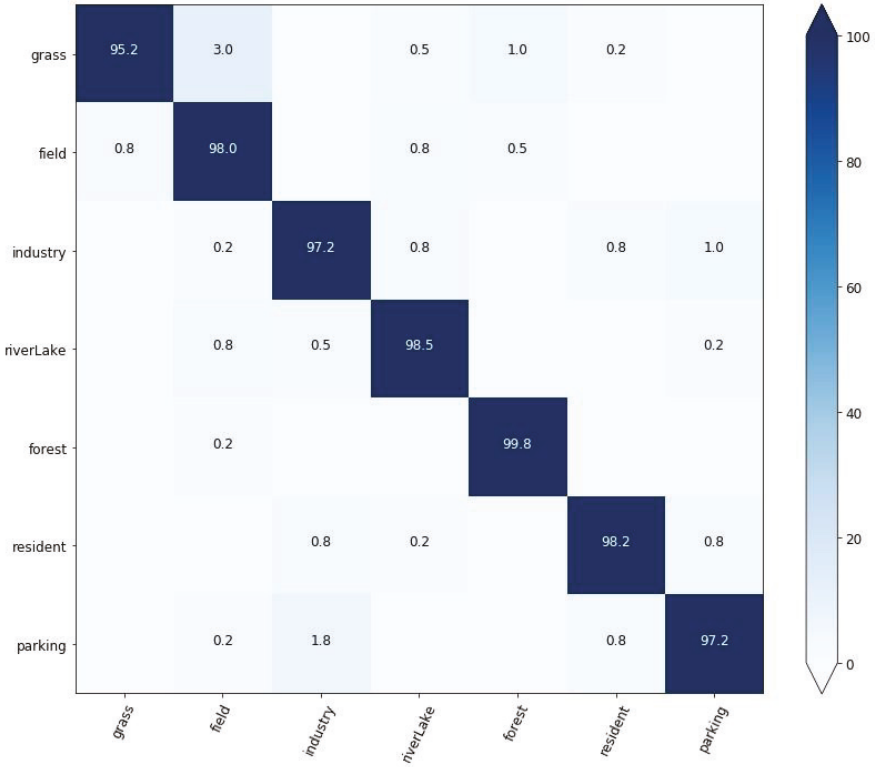


Figure 16. Classification confusion matrix of our method on RSSCN7 dataset.

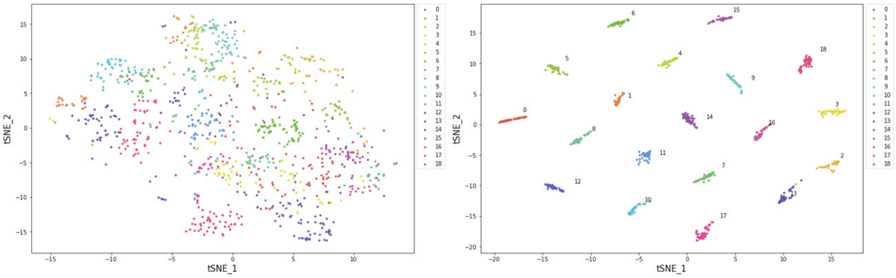
Table 2 shows a comparison of the results obtained for the RSSCN7 dataset classification using different recently proposed methods, including the one proposed in this paper. The proposed two-stage cyclical learning-rate training method achieved the best overall accuracy for two different training ratios. With a 50% training ratio, it produced an increase in accuracy of about 3% compared with other methods.

Table 2. Comparison of the overall accuracy and standard deviations using 50% and 20% training ratios on RSSCN7 dataset.

Method	50% Training Ratio	20% Training Ratio
DBN [20]	77.00	NA
GoogLeNet [33]	85.84 ± 0.92	82.55 ± 1.11
CaffNet [33]	88.25 ± 0.62	85.57 ± 0.95
VGG-16 [33]	87.18 ± 0.94	83.98 ± 0.87
Deep Filter Banks [42]	90.4 ± 0.6	NA
GCFs + LOFs [32]	95.59 ± 0.49	92.47 ± 0.29
RSSCNet (this paper)	97.41 ± 0.27	93.51 ± 0.51

3. Classification of WHU-RS19 dataset

The t-SNE analysis method was used to extract the deep features of the proposed model and to analyze it. In this section, the classification results obtained by applying this model to the WHU-RS19 dataset are discussed. As shown in Figure 17b, as a result of the training, the features became highly clustered, showing that the proposed model helps to improve the classification of the scene.



(a)

(b)

Figure 17. Visual analysis on WHU-RS19 dataset using t-SNE: (**a**) before training; (**b**) after training.

Figure 18 shows the confusion matrix for the WHU-RS19 dataset extracted using VGG16 with the classification types proposed in this study and the optimal classification weights from the training early termination strategy. The training rate used was 60%. The matrix contains the individual classification results for the 19 categories in the dataset. Among these categories, the combinations football field and park and of forest and mountain were the most easily confused during the classification. Table 3 shows a comparison between the results of the WHU-RS19 dataset classification obtained using the proposed method and methods proposed in other recent papers. Using the two-stage cyclical learning-rate training method proposed in this study, our proposed method achieved the best overall accuracy. The kappa coefficients of the WHU-RS19 dataset were 0.9968 and 0.9874 at 60% and 40% training, respectively.

Table 3. Comparison of the overall accuracy and standard deviations using 60% and 40% training ratios on WHU-RS19 dataset.

Method	60% Training Ratio	40% Training Ratio
GoogLeNet [33]	94.71 ± 1.33	93.12 ± 0.82
CaffNet [33]	96.24 ± 0.56	95.11 ± 1.20
VGG-16 [33]	96.05 ± 0.91	95.44 ± 0.60
TEX-Net-LF [36]	98.00 ± 0.52	97.61 ± 0.36
Two-Stream Fusion [38]	98.92 ± 0.52	98.23 ± 0.56
RSSCNet (this paper)	99.46 ± 0.21	98.54 ± 0.37

From the confusion matrix classification results shown in Figure 18, it can be seen that the categories that are misclassified in the WHU-RS19 dataset include “residential”, “forest”, “farmland”, and “bridge” (Figure 19a). We first used the LIME analysis on the misclassified four images and generated the super-pixel feature regions that the model was most interested in (Figure 19b). By observing the super-pixel area in Figure 19b, we can understand why the model misclassifies “residential” as “industrial”, “forest” as “park”, “farmland” as “river”, and “bridge” as “pond”.

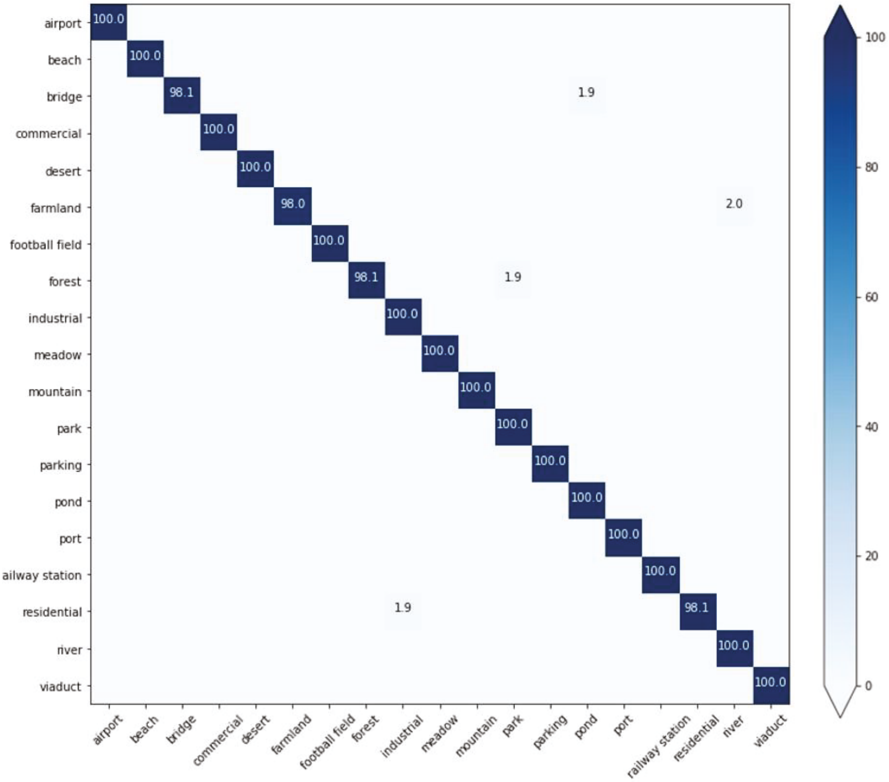


Figure 18. Classification confusion matrix of our method on WHU-RS19 dataset.

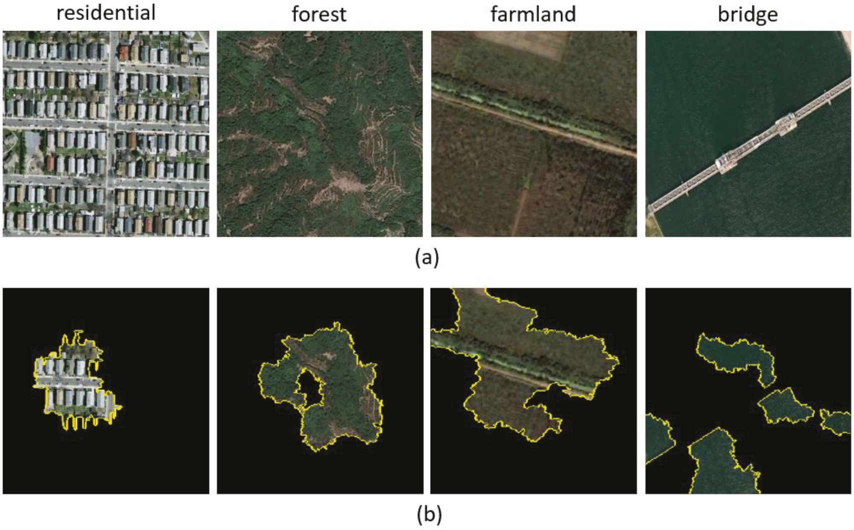


Figure 19. Misclassified images on WHU-RS19 dataset: (**a**) original image; (**b**) super-pixel explanation by LIME analysis.

This research attempted to correct the four images misjudged by the model in Figure 19a with the hopes of improving the model classification performance. First, with regard to the reason for the wrong judgment of the "residential" image, we believe that the other images of the residential category in the dataset contained various bright colors as a whole, while the roofs of the buildings in industrial areas tended to be mostly white. The house colors tended to be white, which may have led to the classification errors. Therefore, the color of the "residential" image was increased in saturation to make it more similar to the other "residential" images in the dataset. From the super-pixel area of the "forest", the cut block contained a part of the bare land, which was different from the other images in this category, which mostly only contained forests. The colors and the details were also blurred. Therefore, the color contrast was enhanced, the overall sharpness of the image was increased, the shadows between the trees were intensified, and the image was prevented from being judged as a "park" again. The image of the "farmland" depicts that the light of the horizontal road in the image is quite obvious, and a green straight line is included in the captured image features. Therefore, we reduced the brightness of the strong part of the image. In the "farmland" parts, we performed a small sharpening to try to remove the noise in the image and strengthen the details of the interval between farmlands. In the last "bridge" image, the feature did not contain the "bridge" feature at all. Therefore, we increased the brightness of the "bridge" itself and the color saturation and sharpness of the image. We also tried to increase the chance of a "bridge" edge being captured as a feature. Figure 20a,b present the four corrected images and their corresponding super-pixel feature regions, respectively. Finally, the four corrected images were replaced with the original images, and the category prediction of the entire dataset was again performed. Consequently, the result reached an overall accuracy of 100%. Figure 21 displays the corrected classification matrix.

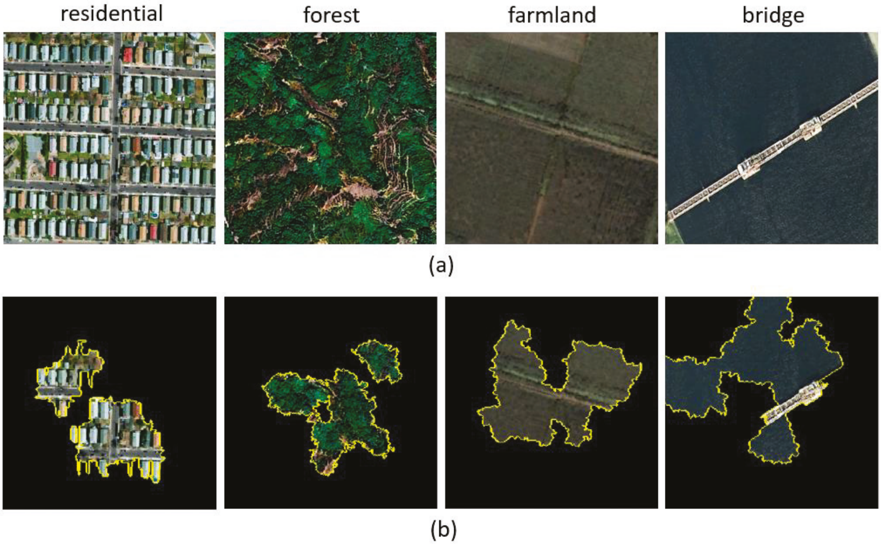


Figure 20. After correction of misclassified images on WHU-RS19 dataset: (**a**) corrected image; (**b**) super-pixel explanation by LIME analysis.

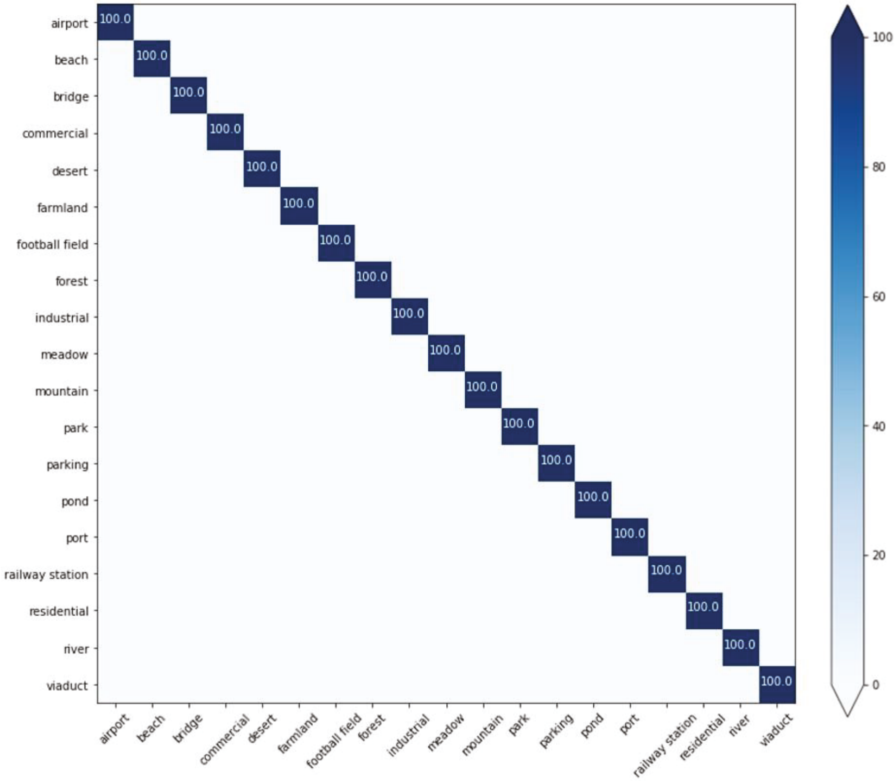


Figure 21. Classification confusion matrix of WHU-RS19 dataset after image correction.

4.2.3. Further Explanation and Discussion

In this section, we further discuss how fine-tuning, a circular learning rate, and an increase in the amount of data can improve the classification performance. The classification results obtained in this study can be expanded to help understand the possible impact of this project on model training.

1. The effectiveness of fine-tuning

In Section 4 of this paper, two-stage training using the proposed model was described, and it was found that the proposed method has significant optimization for training. Moreover, we also wanted to understand, in addition to not carrying out freezing in the first stage, whether freezing the first 19 layers in the second stage would produce different results from those obtained by freezing different layers at two different stages. Therefore, we investigated three different situations: no freezing of any layer, freezing of the top seven layers, and freezing of top the 19 layers; the results for different combinations of these three situations are shown in Figure 22.

The five combinations investigated were no freezing at either stage (shown as "0 + 0"), top seven layers frozen at second stage only ("0 + 4"), top 13 layers frozen at second stage only ("0 + 13"), top 13 layers frozen at first stage only ("13 + 0"), and top 13 layers frozen at two stages ("13 + 13"). From Figure 22, it can be seen that two-stage training with no freezing ("0 + 0") achieved the best testing accuracy. The test accuracy was the worst when the top 13 layers were frozen in the two training phases ("13 + 13"). According to the results of Figure 22, the test accuracy increases as the number of fine-tuning layers increases.

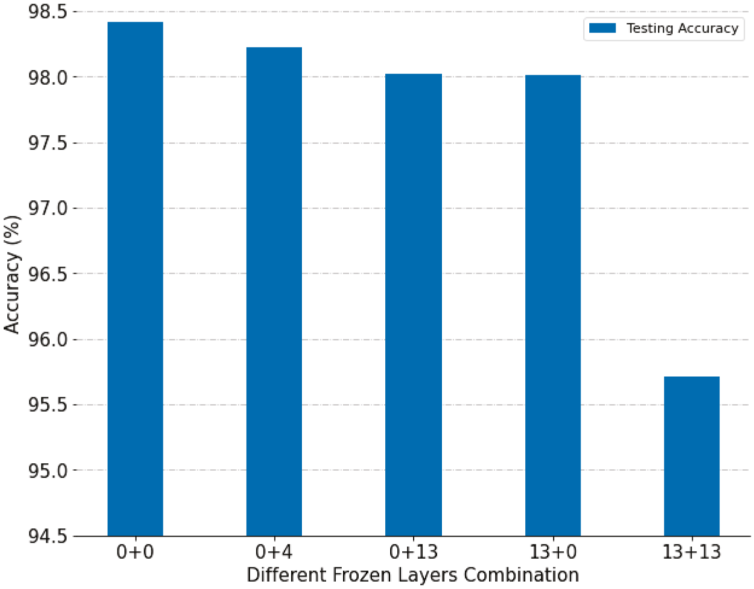


Figure 22. Overall accuracy of fine-tuning using different frozen combinations.

2. Effectiveness of image data augmentation

In this study, after inverting and increasing the number of images by carrying out a small amount of panning and zooming, augmentation training was also included in the training. As shown in Figure 23, doing this also successfully increased the training accuracy. The results here are shown as no data augmentation ((shown as "1-stage" in Figure 23), single-stage data augmentation included in the training ("1-stage DA"), and two-stage data augmentation included in the training ("2-stage DA").

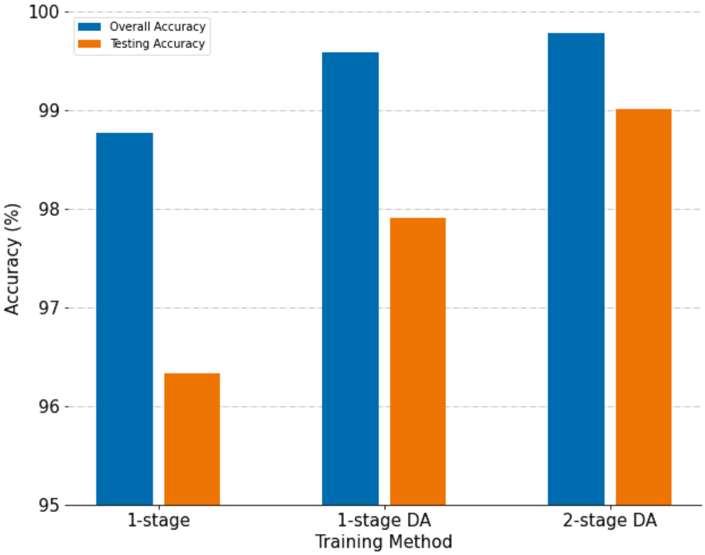


Figure 23. Accuracy of different training methods.

3. Effectiveness of using a two-stage cyclical learning-rate method

In two-stage cyclical learning, the training can be implemented using two different optimizers. In this study, an SGD optimizer with a cyclical learning rate was used in the first stage. In the second stage, an Adam optimizer was used for the training. In two-stage cyclical learning-rate training, this method obtains the best weight in the first stage and then enter the second stage. When used together with the cyclical learning rate, this can greatly accelerate the convergence. Figure 24 shows a comparison of the number of iterations needed to achieve the best level of accuracy for three different training methods.

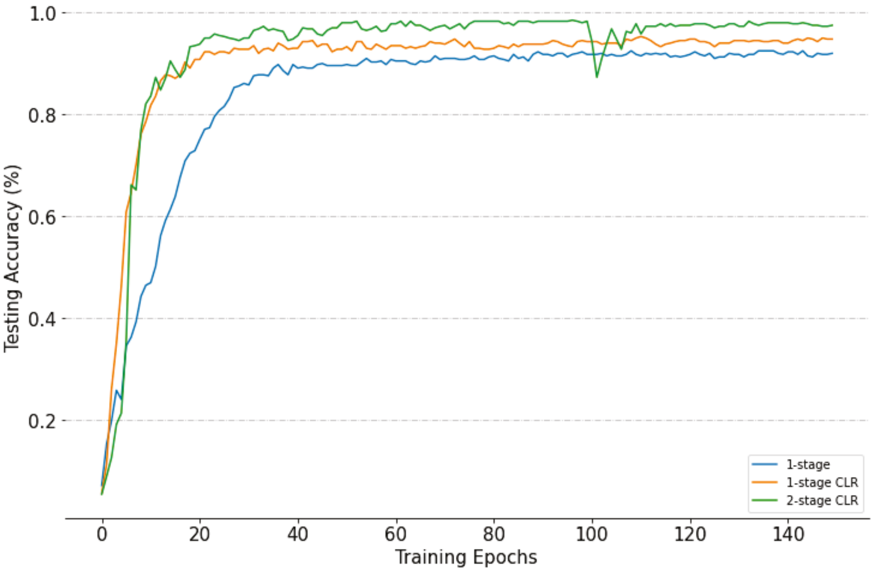


Figure 24. Test accuracy of with and without two-stage cyclical learning-rate method.

When only a single-stage fixed learning rate (shown as "1-stage" in Figure 24) was used for the training, the convergence speed was the slowest and the lowest test accuracy was obtained. The use of a single-stage cyclical learning rate ("1-stage CLR") could speed up the convergence and give a greater chance of avoiding the local optimal solution so that better results could be obtained. When a two-stage circular learning rate ("2-stage CLR") was used in the early part of the second training stage, due to the change of optimizer, the process of finding the best accuracy fluctuated. However, overall, the best accuracy could be reached more quickly, using the smallest number of iterations of the three methods.

5. Conclusions

As a result of continued advances in technology, better-quality and higher-resolution data can be obtained, leading to improvements in remote sensing image classification and in predictions based on it. It takes a lot of time to train and adjust the classification. In order to reduce the time required for the training of the model and to explore how quickly the model can converge with high-generation capability, we recommend the RSSCNet model integrated with the simultaneous use of a two-stage cyclical learning-rate training policy and the no-freezing transfer learning technology that requires only a small number of iterations. In this way, an excellent level of accuracy can be obtained. Data augmentation technology, regularization, and early stopping strategy can then be used to also deal with the problem of limited generalization encountered in the rapid training of deep neural networks. The experimental results that were obtained also confirm that the use of the model and training strategies proposed in this paper can outperform current models in terms of accuracy.

In this study, by using the LIME super-pixel explanation, the root causes of model classification errors were made clearer and a better understanding was obtained. This made it easier to carry out subsequent processing and adjustment of the data or models. After the image correction preprocessing on the four misclassified images using the RSSCNet model in the WHU-RS19 dataset, this image correction procedure was found to improve the overall classification accuracy. This investigation is only a preliminary study.

In future research, we will try to establish universal image correction preprocessing for the case of suspected outliers and merge different XAI (explainable artificial intelligence) analysis technologies to improve interpretation capabilities so that they can be applied to a more diverse range of imagery with different classification issues.

Author Contributions: M.-H.T. and S.-C.H. conceptualized and designed the whole framework and the experiments, as well as wrote the manuscript. S.-C.H. performed the experimental analysis. M.-H.T. contributed to the discussion of the experimental results. H.-C.W. helped to organized and revise the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, grant numbers MOST 108-2621-M-040-002 and MOST 109-2121-M-040-001. The support is greatly appreciated.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gu, Y.; Wang, Y.; Li, Y. A survey on deep learning-driven remote sensing image scene understanding: Scene classification, scene retrieval and scene-guided object detection. *Appl. Sci.* **2019**, *9*, 2110. [CrossRef]
2. Scholl, V.M.; Cattau, M.E.; Joseph, M.B.; Balch, J.K. Integrating national ecological observatory network (neon) airborne remote sensing and in-situ data for optimal tree species classification. *Remote Sens.* **2020**, *12*, 1414. [CrossRef]
3. Cheriyadat, A.M. Unsupervised feature learning for aerial scene classification. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 439–451. [CrossRef]
4. Mahdianpari, M.; Granger, J.E.; Mohammadimanesh, F.; Salehi, B.; Brisco, B.; Homayouni, S.; Gill, E.; Huberty, B.; Lang, M. Meta-analysis of wetland classification using remote sensing: A systematic review of a 40-year trend in north america. *Remote Sens.* **2020**, *12*, 1882. [CrossRef]
5. Luus, F.P.S.; Salmon, B.P.; Van den Bergh, F.; Maharaj, B.T.J. Multiview deep learning for land-use classification. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2448–2452. [CrossRef]
6. Maire, F.; Mejias, L.; Hodgson, A. A convolutional neural network for automatic analysis of aerial imagery. In Proceedings of the 2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Wollongong, New South Wales, Australia, 25–27 November 2014; pp. 1–8.
7. Wang, T.; Thomasson, J.A.; Yang, C.; Isakeit, T.; Nichols, R.L. Automatic classification of cotton root rot disease based on uav remote sensing. *Remote Sens.* **2020**, *12*, 1310. [CrossRef]
8. Tong, X.-Y.; Xia, G.-S.; Lu, Q.; Shen, H.; Li, S.; You, S.; Zhang, L. Land-cover classification with high-resolution remote sensing images using transferable deep models. *Remote Sens. Environ.* **2020**, *237*, 111322. [CrossRef]
9. Zhang, Z.; Jiang, R.; Mei, S.; Zhang, S.; Zhang, Y. Rotation-invariant feature learning for object detection in vhr optical remote sensing images by double-net. *IEEE Access* **2019**, *8*, 20818–20827. [CrossRef]
10. Zhong, Y.; Han, X.; Zhang, L. Multi-class geospatial object detection based on a position-sensitive balancing framework for high spatial resolution remote sensing imagery. *ISPRS J. Photogramm. Remote Sens.* **2018**, *138*, 281–294. [CrossRef]
11. Hu, F.; Xia, G.-S.; Hu, J.; Zhang, L. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sens.* **2015**, *7*, 14680–14707. [CrossRef]
12. Nogueira, K.; Penatti, O.A.; dos Santos, J.A. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognit.* **2017**, *61*, 539–556. [CrossRef]

13. Penatti, O.A.; Nogueira, K.; Dos Santos, J.A. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 44–51.
14. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.
15. Smith, L.N.; Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In Proceedings of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, Baltimore, MD, USA, 10 May 2019; p. 1100612.
16. Leclerc, G.; Madry, A. The two regimes of deep network training. *arXiv* **2020**, arXiv:2002.10376.
17. Caruana, R.; Lawrence, S.; Giles, C.L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In Proceedings of the Advances in Neural Information Processing Systems, Cambridge, MA, USA, 2001; pp. 402–408.
18. Sheng, G.; Yang, W.; Xu, T.; Sun, H. High-resolution satellite scene classification using a sparse coding based multiple feature combination. *Int. J. Remote Sens.* **2012**, *33*, 2395–2412. [CrossRef]
19. Yang, Y.; Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010; pp. 270–279.
20. Zou, Q.; Ni, L.; Zhang, T.; Wang, Q. Deep learning based feature selection for remote sensing scene classification. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2321–2325. [CrossRef]
21. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
22. Perez, H.; Tah, J.H.; Mosavi, A. Deep learning for detecting building defects using convolutional neural networks. *Sensors* **2019**, *19*, 3556. [CrossRef] [PubMed]
23. Clevert, D.-A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289.
24. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
25. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
26. Maaten, L.V.D.; Hinton, G. Visualizing data using t-sne. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
27. Linderman, G.C.; Rachh, M.; Hoskins, J.G.; Steinerberger, S.; Kluger, Y. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nat. Methods* **2019**, *16*, 243–245. [CrossRef] [PubMed]
28. Song, W.; Wang, L.; Liu, P.; Choo, K.-K.R. Improved t-sne based manifold dimensional reduction for remote sensing data processing. *Multimed. Tools Appl.* **2019**, *78*, 4311–4326. [CrossRef]
29. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
32. Zeng, D.; Chen, S.; Chen, B.; Li, S. Improving remote sensing scene classification by integrating global-context and local-object features. *Remote Sens.* **2018**, *10*, 734. [CrossRef]
33. Xia, G.-S.; Hu, J.; Hu, F.; Shi, B.; Bai, X.; Zhong, Y.; Zhang, L.; Lu, X. Aid: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3965–3981. [CrossRef]
34. Bian, X.; Chen, C.; Tian, L.; Du, Q. Fusing local and global features for high-resolution scene classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 2889–2901. [CrossRef]
35. Li, P.; Ren, P.; Zhang, X.; Wang, Q.; Zhu, X.; Wang, L. Region-wise deep feature representation for remote sensing images. *Remote Sens.* **2018**, *10*, 871. [CrossRef]

36. Anwer, R.M.; Khan, F.S.; van de Weijer, J.; Molinier, M.; Laaksonen, J. Binary patterns encoded convolutional neural networks for texture recognition and remote sensing scene classification. *ISPRS J. Photogramm. Remote Sens.* **2018**, *138*, 74–85. [CrossRef]
37. Liu, B.-D.; Xie, W.-Y.; Meng, J.; Li, Y.; Wang, Y. Hybrid collaborative representation for remote-sensing image scene classification. *Remote Sens.* **2018**, *10*, 1934. [CrossRef]
38. Yu, Y.; Liu, F. A two-stream deep fusion framework for high-resolution aerial scene classification. *Comput. Intell. Neurosci.* **2018**, *2018*, 8639367. [CrossRef]
39. Huang, H.; Xu, K. Combing triple-part features of convolutional neural networks for scene classification in remote sensing. *Remote Sens.* **2019**, *11*, 1687. [CrossRef]
40. Zhang, W.; Tang, P.; Zhao, L. Remote sensing image scene classification using cnn-capsnet. *Remote Sens.* **2019**, *11*, 494. [CrossRef]
41. Guo, Y.; Ji, J.; Shi, D.; Ye, Q.; Xie, H. Multi-view feature learning for vhr remote sensing image classification. *Multimed. Tools Appl.* **2020**. [CrossRef]
42. Wu, H.; Liu, B.; Su, W.; Zhang, W.; Sun, J. Deep filter banks for land-use scene classification. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 1895–1899. [CrossRef]

applied sciences

MDPI

Article

Comparing Classical and Modern Machine Learning Techniques for Monitoring Pedestrian Workers in Top-View Construction Site Video Sequences

Marcel Neuhausen, Dennis Pawlowski * and Markus König

Computing in Engineering Department, Ruhr-University Bochum, 44801 Bochum, Germany;
marcel.neuhausen@ruhr-uni-bochum.de (M.N.); koenig@inf.bi.ruhr-uni-bochum.de (M.K.)
* Correspondence: dennis.pawlowski@ruhr-uni-bochum.de

Received: 29 October 2020; Accepted: 24 November 2020; Published: 27 November 2020

Abstract: Keeping an overview of all ongoing processes on construction sites is almost unfeasible, especially for the construction workers executing their tasks. It is difficult for workers to concentrate on their work while paying attention to other processes. If their workflows in hazardous areas do not run properly, this can lead to dangerous accidents. Tracking pedestrian workers could improve the productivity and safety management on construction sites. For this, vision-based tracking approaches are suitable, but the training and evaluation of such a system requires a large amount of data originating from construction sites. These are rarely available, which complicates deep learning approaches. Thus, we use a small generic dataset and juxtapose a deep learning detector with an approach based on classical machine learning techniques. We identify workers using a YOLOv3 detector and compare its performance with an approach based on a soft cascaded classifier. Afterwards, tracking is done by a Kalman filter. In our experiments, the classical approach outperforms YOLOv3 on the detection task given a small training dataset. However, the Kalman filter is sufficiently robust to compensate for the drawbacks of YOLOv3. We found that both approaches generally yield a satisfying tracking performances but feature different characteristics.

Keywords: cascaded classifier; computer vision; construction site management; deep learning; tracking

1. Introduction

Construction sites constitute highly dynamic environments in which workers execute diverse orders simultaneously. Workers need to perform tasks, interact with heavy construction equipment and keep an eye on their surroundings, which is difficult for complex tasks. This requires construction workers to have a high level of concentration to avoid mistakes. If the construction site is noisy and congested, it can be difficult for the construction workers to concentrate on their work and the environment at the same time. In addition, the continuous change of a construction site often leads to hazardous situations. Heavy construction machines move across the site to execute their jobs. Construction vehicles cross the worker's paths and cranes lift loads over their heads. In addition, pedestrian workers inevitably share the same workspaces with construction machines or interact with them in order to accomplish their orders [1]. As a result, worker's activities often happen in close proximity to heavy machinery. Hazardous situations such as close calls can occur as a consequence of this [2]. Furthermore, in certain cases, people can misjudge the danger. Due to the mentioned facts, it is to be expected that workflows on construction sites are not always ideal. In addition, the incidents lead to hazardous situations for pedestrians on construction sites.

Thereby, the pedestrian worker can be injured or have a fatal accident. For these reasons, construction workers undergo regular training to raise their awareness with respect to hazardous situations [3] as well as to develop their knowledge and skills [4] to improve their workflows. Despite this effort, working in the surroundings of heavy construction machines remains to be a hazardous job. Identifying a reasonable workflow during its execution in a steadily changing environment also keeps to be a challenging concern. Accordingly, productivity and safety problems continue to occur on construction sites.

Monitoring pedestrian workers on the sites from a top-view perspective could improve this situation. This way, the worker's trajectories could be analyzed and adaptations to their workflows could be made during their work [5]. In addition, a monitoring system would help the machine operators in their work, as they usually do not have a complete overview of the environment [1]. In this case, the positions and movement directions of workers who are nearby could be provided to the machine operators. This would allow them to recognize hazardous situations at an early stage and take appropriate action to prevent an accident. The mentioned scenarios represent possible applications of a monitoring system on construction sites, for which a suitable method has to be investigated. Therefore, in this paper, we only focus on the detection and tracking of construction workers, since this is the basis for a surveillance system.

Different approaches have already been made in tracking pedestrian workers on construction sites. Depending on the surrounding, various technologies are used for monitoring workers in construction related scenarios [6]. The literature often refers to global navigation satellite system (GNSS) tags for outdoor localization on construction sites [7,8]. This is mainly applied to track construction machines and equipment, but some approaches make use of GNSS to locate pedestrians on site [9,10]. Near buildings, walls or large construction elements, GNSS-based localization is affected by multipath effects caused by reflections of the signal on these objects [11,12]. Since construction workers often work near these objects, tracking such workers with GNSS becomes unreliable. Less sensitive approaches rely on radio-frequency technologies. These also include the attachment of tags to the worker's gear. Corresponding readers can either be stationary on the construction site or attached to construction machinery. Employing a setup that utilizes radio-frequency identification (RFID) technology enables the warning of workers and machine operators whenever a tagged worker gets into the range of a machine's reader [13] or specific zones [14]. Other approaches develop systems noticing pedestrian workers when entering certain zones by localizing them using ultra-wideband (UWB) tags [7]. Although the accuracy can be improved by combining RFID technology with ultrasound [15], a precise localization and tracking of workers in real time constitutes a challenging task [7] which has not been sufficiently solved in general yet [16]. Besides technical deficiencies, the deployment of such a system for the implementation of a tracking application on a real construction site requires a tag for each worker. Additionally, at least three receivers per monitored area have to be installed in the case of two-dimensional tracking which results in high acquisition costs [6]. Furthermore, workers perceive the attachment of tags with unique identifiers (IDs) to their gear to be obtrusive and to cause discomfort [17].

In contrast to tag-based methods, camera-based tracking approaches provide cost-effective surveillance alternatives. The costs amount to one camera per monitored area and there are no further costs for additional equipment for the workers. In research, some effort has already been made to detect construction worker's in camera images [18]. Park and Brilakis [8] built a real-time capable detection scheme to recognize construction workers in camera images for initializing a visual tracker. They used background subtraction via a median filter to find moving objects within the images. Then, a pedestrian detection is performed on these objects using a support vector machine (SVM) which operates on Histogram of Oriented Gradients (HOG) features. Exploiting the loud colors of the worker's safety vests, construction workers are identified from the pedestrian detections by clustering hue, saturation, value (HSV) color histograms using a k-Nearest Neighbors (k-NN) algorithm. Chi and Caladas [19] also decided on background subtraction to identify moving objects before classifying these by a neural network approach. In [20], the background subtraction is exchanged by a sliding window approach. However, similar to Park and Brilakis [8], they used HOG and color features but concatenate them to a single vector which is passed to a SVM in order to identify workers. Using color and spatial models classified by a Gaussian kernel, Yang et al. [21] decided on a similar strategy. In recent years, deep neural network based approaches have become prominent. To recognize worker's activities, Luo et al. [22] applied temporal segment networks. Son et al. [23] used an region based convolutional neural network (R-CNN) based on ResNet to detect workers under changing conditions. Luo et al. [24] proposed an approach which detects construction workers in oblique images of cameras mounted in heights. They applied YOLO for detection but only achieved a precision of about 75%. Vierling et al. [25] proposed a convolutional neural network (CNN)-based concept detecting workers in top-view images. To cope with high altitudes, their approach relies on several zoom levels each with a separate CNN for detection. An additional meta CNN is used to choose the correct zoom level for a certain height.

Despite the use of transfer learning techniques, CNN-based approaches usually require large amounts of training data. Corresponding data, which represent construction sites from a top view perspective, are rarely available. In addition, the generation of a sufficiently large dataset is a time-consuming and demanding task. Beyond that, these networks commonly operate on small image sizes of about 400 × 400 px. High resolution images cannot be processed in real time without disproportionately large amounts of computational power or without drastically downscaling images. Since surveillance cameras monitor large areas of construction sites, workers occupy only very small parts of the image. Besides the fact that detecting small objects with CNNs is a challenge, reducing the size of the image makes detection even more difficult. The downscaling may eliminate relevant features in the image required for a reasonable detection.

As it is unclear whether CNNs can outperform classical machine learning methods on these terms, we conduct a comparison in this paper. The goal of the comparison is to find out whether one of the two approaches can satisfactorily track several construction workers when the amount of training data is small. In doing so, we restrict our focus to one view within one construction site at first. In this case, a camera could be mounted, e.g., on the mast of a tower crane. Alternatively, several cameras could be used to completely monitor a construction site, but this is not covered in this paper. Because no suitable dataset is publicly available for comparison, we assemble a small generic dataset ourselves. This shows pedestrian construction workers from a top view perspective under different conditions. As a representative CNN approach, we choose YOLOv3 [26], since it is a state-of-the-art detection network. For its counterpart from the field of classical machine learning, we rely on preliminary work [27] discussing diverse computer vision techniques for monitoring construction workers. In this work, we juxtapose eligible methods and develop a theoretical concept relying on a classical machine learning method. Based on these results, we implement a tracking approach based on a soft cascaded classifier in the course of this paper. A simple Kalman filter is applied to both approaches in order to track the detected workers within the recorded video sequence. In our experiments, we compared the detection and tracking results of our implemented approach with those of YOLOv3 trained on the same data. We found that our classical machine learning approach yields substantial better detection results on the small dataset than the CNN. However, the Kalman filter proves to be sufficiently robust to compensate for the lower detection quality of YOLOv3. Owing to this, both approaches perform similarly well on the tracking of pedestrian workers in general. Nevertheless, each approach possesses different tracking characteristics. A general recommendation can, thus, not be made. The appropriate tracking solution has to be determined with respect to the demands of the particular application.

2. Materials and Methods

To compare the performance of CNN-based and classical computer vision approaches to the monitoring of pedestrian workers, we assembled a small characteristic dataset. This dataset includes different scenarios as well as various environmental conditions. Section 2.1 describes the dataset in detail. This dataset is used for the training and the evaluation of both approaches chosen for our comparison. The classical approach is composed of a soft cascaded classifier and a background subtraction which preprocesses the input images to enable detection in real-time. Its detailed structure and the parameter optimization are described in Section 2.2. YOLOv3 also possesses several hyperparameters. In Section 2.3, we elaborate on our choice of those hyperparameters. For a better comparison, the detections of both approaches are tracked by the same method over the course of the video sequences. We chose Kalman filtering for this as it is a simple but robust method which is sufficient for the purpose of comparing the two approaches. Details about the Kalman filter's motion model and other required parameters are given in Section 2.4. The implementation and testing of the two approaches was done with the programming language C++ on a standard computer.

2.1. Dataset Acquisition

Top-view scenes of construction sites have rarely been recorded. Accordingly, a dataset reasonable for our purpose has not been published yet. For that reason, we recorded video sequences to train and test our approach explicitly for the scope of this paper. It is important to know if different backgrounds and lighting conditions can have an influence on the tracking of construction workers. In addition, we want to test if our approach can distinguish between different moving objects. The videos were therefore taken in two scenarios with different levels of difficulty for our approach: in the first one, construction workers act on a uniformly plastered terrain, whereas a mixture of gravelled and plastered areas is chosen for the second scenario. While the first scene is illuminated well, the second scene is slightly overexposed. Both sequences are recorded by a non-pivoting camera at a height of 20 m, which is aligned to the ground and has a fixed position (see Figure 1). This results in a top-view perspective in the center of the image, but becomes oblique at the borders of the image. In accordance with a typical crane camera set up [28], all videos were recorded with a frame rate of 25 fps and a resolution of 1920 × 1080 px. In both scenarios, construction workers wearing safety vests and helmets walk randomly through the camera's field of view including sudden stops and directional changes. They also interact with static construction specific elements such as pylons and barrier tapes. Of course, exposing workers intentionally to hazardous situations or heavy construction machinery is unwarrantable. Such hazardous situations are substituted by smaller moving vehicles instead which still allows for evaluating the correct classification of moving objects. In both video sequences, the construction workers are manually labeled by hand in order to prepare both the ground truth and the training data. Rectangular areas surrounding the worker's heads and shoulders are used to indicate the positions of the workers.

Datasets for training, validation and evaluation are generated from the labeled sequences. For this, we divide each sequence into three shares of equal length. From the first share of both sequences, we generate the training dataset. This training dataset includes 1000 pedestrian construction worker samples (see Figure 2). Samples are only extracted from every 10th frame to reduce the similarity among the samples. For generating the validation dataset, we proceed analogously with the second share of both sequences. The validation dataset also contains 1000 pedestrian construction worker samples. From the third part, we generate evaluation data that consist of video sequences with an average length of 12 s. From each sequence, we choose a representative scene which contains settings commonly occurring on construction sites. Both scenes show up to four pedestrian construction workers simultaneously and other objects colored similarly to the worker's safety vests that are either static or moving. The static objects are red and white barrier posts, barrier tapes and a red barrier on a gravel area. These elements have similarities to the colors of the construction workers. Moving objects include a red vehicle that moves linearly in the same direction as a construction worker. In addition, the red color between the worn safety vest and the vehicle is similar. The workers walk through the scene while sometimes passing each other closely. In addition to these two scenes, we choose a third one to evaluate our approach with regard to moving vehicles. This scene shows a single worker walking while a car approaches him from behind.

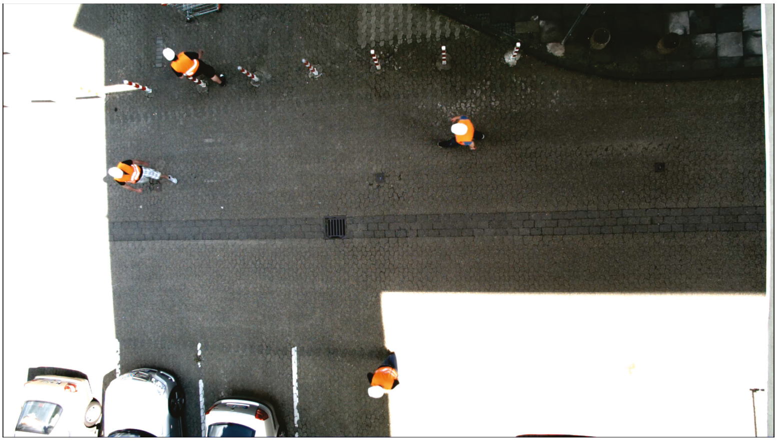


Figure 1. The construction workers are captured by a stationary camera which is located 20 m above the ground. The camera is pointed at the ground.

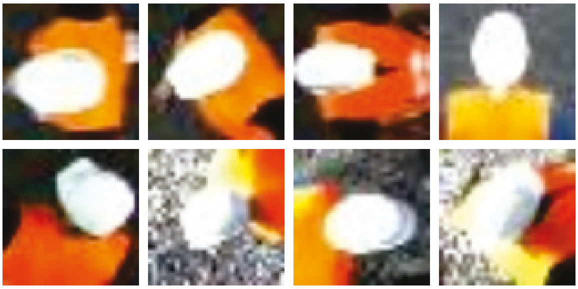


Figure 2. The training dataset contains construction workers taken from the top view. The workers are in different orientations and illuminated in different ways.

2.2. Classical Detection Approach

Based on our theoretical concept previously proposed in [27], we implemented an explicit approach to detection of construction workers in top view images. As shown in Figure 3, detection is done by first extracting relevant regions of interest (RoIs) from the current camera image. Afterwards, each region is classified to determine whether it contains a worker or not.

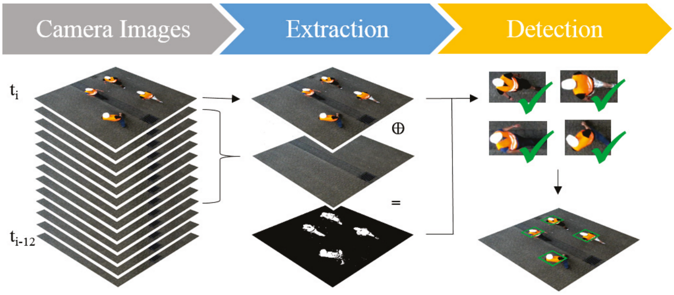


Figure 3. Concept of the classical detection approach. Each time step t corresponds to an image frame. The current frame in time step t_i is subtracted from a background model which is generated from images of the previous time steps. This results in foreground blobs which represent RoIs. Image patches of the current image corresponding to these regions are fed into the classifier determining the presence of construction workers.

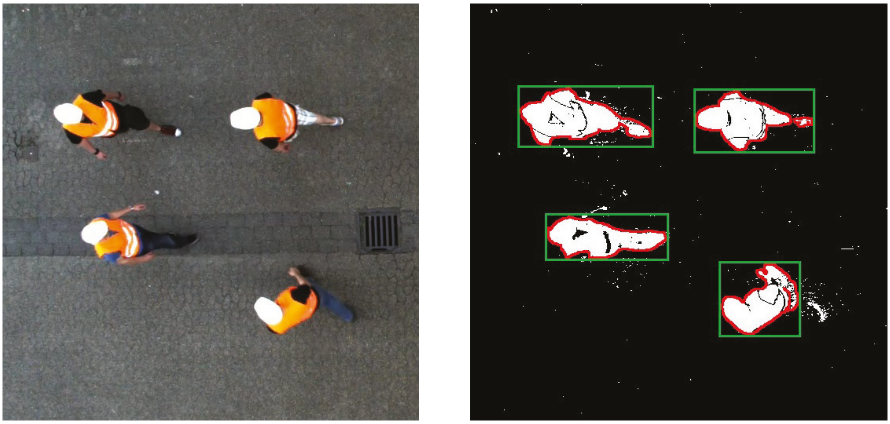
Most previously implemented image-based approaches in civil engineering are based on a frontal camera perspective. In our case, however, the camera is mounted at a height in order to monitor a large area and to satisfy the requirements of different applications. This leads to a top-view perspective. Most features that make up a pedestrian, such as legs and arms, are usually covered by his own body due to the perspective. Although there are already approaches for the general detection of pedestrians in top-view images, detecting pedestrian workers can be eased by leveraging typical features that characterize those workers. On construction sites, workers wear helmets and safety vests with loud colors that constitute clearly visible and prominent features. Similar to the findings of Park and Brilakis [8], a combination of motion, color and shape features is more reasonable for a proper detection in our scenario than the use of common pedestrian detection features. While motion is a fundamental characteristic of pedestrians, it is not unique to them. Other objects such as construction vehicles may also move through the cameras field of view so that classification by motion is unrewarding. Notwithstanding, constraining the detection to image regions containing movement spares the expense of investigating the entire image. This improves the subsequent detection in two different ways. On the one hand, applying the classifier to only a few regions of the image significantly speeds up the detection process which allows for the monitoring of substantially larger areas of a site without an appreciable time lag. On the other hand, this preselection excludes most image regions not containing any construction workers from the further detection process which highly reduces false positive detections in advance.

To find regions of motion, moving foreground objects have to be separated from the background. For this, Gaussian mixture models are frequently used and well established methods estimating a scene's background are available [8]. We use an improved adaptive Gaussian mixture model (GMM) approach [29], because it is insensitive to background movements that often occur in outdoor scenarios, e.g., wobbling bushes or leaves blown by the wind [30]. To adapt to changes in the scene such as varying illumination conditions or newly positioned and removed static objects, the mixtures of Gaussians are learned and adjusted over time. Accordingly, the Gaussians' parameters, their number per pixel, as well as the learning rate are updated online by the adaptive approach while applying the model to consecutive video frames.

Comparing the current frame (see Figure 4a) to the learned background model results in a binary segmentation image which indicates fore- and background pixels. In the next step, all connected foreground pixels are aggregated to regions of motion by blob detection. For this purpose, an algorithm is used that finds contours in the binary image [31]. In a further step, rectangles are formed, which enclose each contour. These rectangles identify the RoIs for the further detection process, as shown in Figure 4b.

Since the background subtraction identifies the relevant areas within the image which are likely to contain pedestrian construction workers, the detector only has to focus on those image regions. Each of those RoIs contains a single moving object in the scene. Distinguishing between a worker and any other object can be done by a binary classification for each RoI.

As described above, construction workers in top-view images are characterized best by their motion, color and shape. The classification of color and shape features provides a proper basis of decision-making since we already use motion to find candidate regions. Following this, we use color histograms as they are simple but effective color feature descriptors. The histograms are computed on the hue and saturation channels of the HSV color space. Since the value channel decodes brightness, it is neglected in favor of the histograms' invariance to changes in illumination conditions [8], which is a common issue in outdoor scenes. For determining shape information, we decided on Haar-like features. The choice of these features allows us to design both feature descriptors as low-order integral channel features [32]. This guarantees the efficient computation of the feature responses which increases the detection speed. However, it remains unclear how to arrange those feature descriptors' positions and sizes on an image patch so that they optimally respond to a construction worker.



(**a**) Scene

(**b**) Background image

Figure 4. Background subtraction example: (**a**) video frame of walking construction workers; and (**b**) background image with white foreground pixels aggregated to blobs of motion (red) and the corresponding RoIs (green) for further processing.

For this reason, we apply a Soft Cascaded Classifier [33] which learns the optimal set of features using AdaBoost [34]. For this, we deduce weak classifiers from the feature descriptors by thresholding their responses. Then, we generate a weak classifier pool containing thresholded feature descriptors at all conceivable sizes and positions in an image patch. During training, AdaBoost iteratively draws that weak classifier from the pool which separates the set of samples best. This way, a strong classifier emerges from the set of chosen weak classifiers. Figure 5 visualizes the process by the example of the first five Haar-like feature descriptors chosen by AdaBoost.

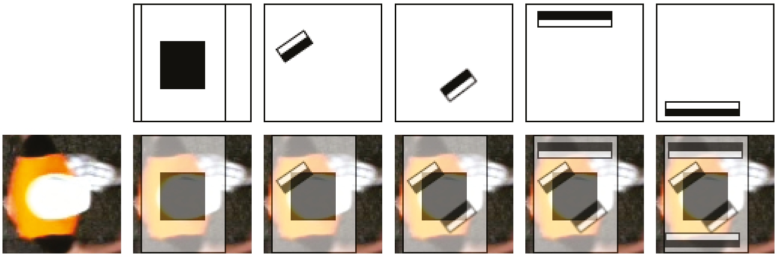


Figure 5. Iterative feature selection by AdaBoost (from left to right). A shape model is learned from Haar-like features. The first three iterations approximate the worker's head while the following two focus on the worker's shoulders.

By a subsequent calibration, the weak classifiers within the strong classifier are arranged into cascading stages. A sample is only passed further along the cascade if the evidence of belonging to the positive class is sufficiently strong. As a result, negative samples are rejected early in the cascade which drastically speeds up the classification process for a vast number of samples to be verified.

2.3. Hyperparameter Setting of YOLOv3

Training YOLOv3 requires the adjustment of several hyperparameters. Despite extensive use of YOLO's built-in data augmentation features, the training dataset is too small for training a reasonable detector from scratch. For this reason, we base our training on the Darknet53 model which has been pre-trained on ImageNet [35]. To train YOLO on pedestrian workers, we pass the 1000 sample images of our training dataset in a mini batch size of 64 to the network. In the beginning, we use a high learning rate of 0.001 in order to quickly adjust the detector towards the new class. Every 3800 epochs, the learning rate is scaled down by a factor of 0.1, facilitating the learning process to converge towards an optimal result. For regularization, we adapt the weights by a momentum of 0.9 and a weight decay of 0.0005. After about 10,400 epochs, the validation error stops decreasing so that the training is stopped.

2.4. Tracking Using Kalman Filtering

Knowing the current position of construction workers as provided by a detector can already be advantageous for productivity management and safety applications. However, this can be further improved by tracking the workers over time. This allows keeping track of entire workflows as well as anticipating the worker's movement directions.

Modern tracking approaches rely on a motion model which predicts an object's trajectory and an appearance model to recognize the tracked object in the following video frames. However, it is sufficient for our purpose to rely on a motion model only. The applied detector compensates for the omitted appearance model. Hereby, the detector implicitly adopts the recognition task.

In this paper, we apply Kalman filtering. Its simplicity and robustness make it a good choice for the comparison of the performance of the two proposed detection methods. The Kalman filter is based on a motion model which only describes the relationship between the tracked object's current state and its predicted state at the next time step. For this, the model consists of the tracked object's current state and a prediction matrix which models the transition from one time step to another. A tracked worker's current state $[x, y, v_x, v_y]$ can be described by the worker's position x, y and his velocity v_x, v_y in x and y directions. To predict the worker's state in the following time step $t + 1$, the prediction matrix is applied to the current state at time step t with a time duration Δt,

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ v_{x,t+1} \\ v_{y,t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ v_{x,t} \\ v_{y,t} \end{pmatrix}. \tag{1}$$

The predicted state is then set to the new current state of the Kalman filter's model. Afterwards, the state can be updated using the actual measurements of the worker's position $[x_t, y_t]$. In our case, the detected construction workers serve as both tracker initializations and observed measurements. The detections are spatially correlated with regard to already existing tracks. Detections with a close spatial proximity are assigned to the individual track. For detected workers not matching a pre-existing track, a new track is set up. The Kalman filter then predicts the location of a worker frame by frame. Detections close to the predicted locations which we assign to the track give evidence to correct the Kalman filter's motion model and prevent the predictions from drifting off.

To determine optimal values for the noise covariance matrices Q and R, we performed a grid search using both detection algorithms. Values on the matrices' diagonals in the range of $[1, 50]$ were considered for this. We found that the optimal values for the diagonals of Q and R are 1 and 50, respectively.

3. Optimization

Although some applications such as workflow optimization are commonly executed asynchronously to the recordings, other applications such as the assistance of machine operators require current worker's trajectories. Accordingly, a reasonable approach to the monitoring of pedestrian construction workers demands real-time capability so that worker's locations trajectories are provided for every video frame given by the camera. To properly compare the CNN-based and classical computer vision approaches, we optimize their detection quality with respect to a sufficient speed with regard to the camera's specification.

The detection speed of the soft cascaded classifier is mainly determined by the number of weak classifiers constituting the strong classifier. They define the depth of the cascade and, thus, the quantity of feature applications required to correctly classify a given sample. Besides, the number of training samples and the features' explicit manifestations situated in the feature pool highly affect this method's detection quality. In Section 3.1, we optimize the soft cascaded classifier with regard to these parameters and discuss the chosen values.

YOLO's detection speed is restricted by a single hyperparameter which is the size of the input images. The larger are the input images, the more convolutions are involved, which results in substantially slower processing speed. Accordingly, Section 3.2 addresses the optimal choice of the input image size for a proper detection with respect to the real-time capability.

3.1. Optimization of the Soft Cascaded Classifier

A considerably high precision of the detector is desirable in order to satisfactorily initialize the trackers by only actual construction workers [8]. This prevents the initialization of false positive tracks as well as tracker updates based on false detections. However, to be suitable for a monitoring application, the detector must be able to identify workers in real-time in the first instance. To obtain a satisfying classifier in terms of detection speed with a preferably high precision at an acceptable recall, we conduct a grid search by varying different training parameters.

In preliminary experiments, we already found that subdividing the color histograms into five bins is sufficient for the classifier to properly recognize the characteristics of a construction worker's helmet and safety vest. Other parameters to be investigated include the minimal size of the feature descriptors provided by the feature pool. Too small feature descriptors may affect the classification. The feature descriptors' sensitivity to noise increases with decreasing size which may impair the classification. Similar results hold for slight translations, rotations and scalings of the object to be detected within the RoI provided by the background subtraction. By this, too small feature descriptors may easily be positioned off the corresponding feature. On the other hand, large feature descriptors may miss small features. For these reasons, we investigate the effect on the feature descriptors' sizes to the detection quality. We vary the minimal feature size of the feature descriptors provided in the feature pool for training between 10% and 30% of the given image patch size. Additionally, we vary the quantity of training samples from 200 to 2000 to find the optimal generalization behavior. Finally, to speed up the recognition process, we determine the minimum number of weak classifiers required for a reliable classification. For that, we vary the number of weak classifiers constituting our cascade from 50 to 350. To compare the detection results while varying a parameter, we juxtapose the classifiers' receiver operating characteristic (ROC) curves. For this, we apply the trained classifiers to the validation dataset and plot their true positive rate (TPR) against their false positive rate (FPR) with respect to the particular classifier's confidence. Figure 6 illustrates the ROC curves for all three parameters. In addition, we calculate the accuracy of the classifier when the number of weak classifiers is varied. During the variation, we apply the classifier to the validation dataset and to the training dataset and plot the accuracy curves. The results are shown in Figure 7.

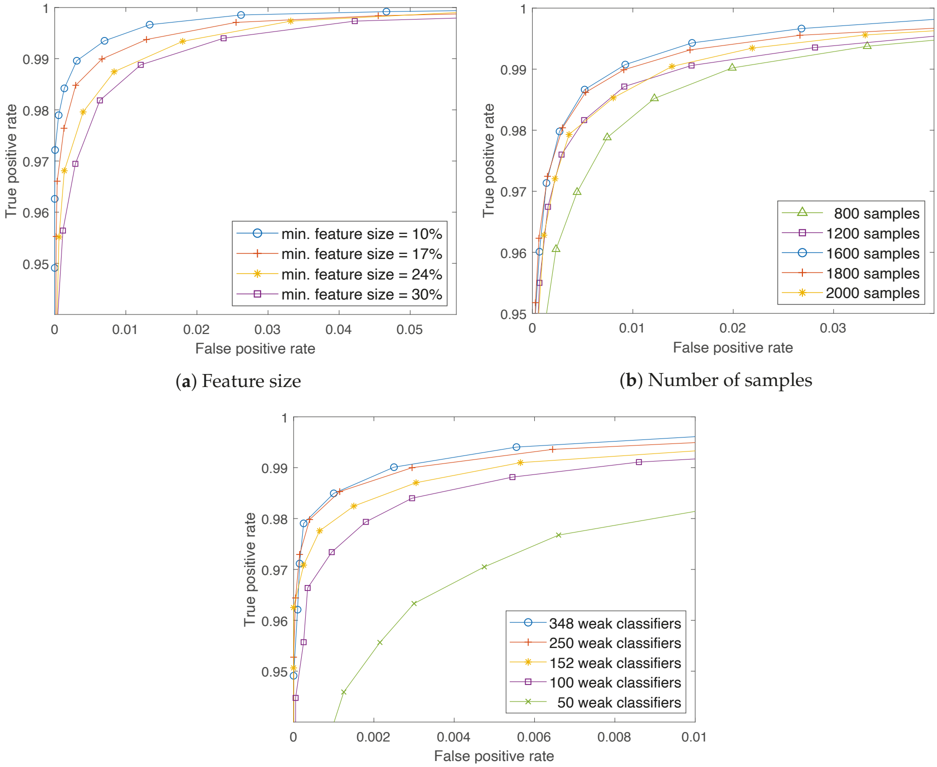
As can be derived from Figure 6a, providing a feature pool in which feature descriptors have a minimal size of 10% of the image patch size for the classifier yields best classification results. The classification quality decreases with increasing minimal feature size. This shows that the classification is not impaired but even improved by rather small feature descriptors. Noise and the worker's positioning seem to have only little effect, if any.

The effect of different amounts of training samples are shown in Figure 6b. As can be seen, the classification quality increases up to a total number of 1600 training samples. From then on, the quality starts to decrease again. This shows that the classifier looses essential features that describe a construction worker in general terms if more than 800 positive and negative samples are used.

As Figure 6a depicts, the general classification quality increases with the number of the cascaded weak classifiers. Nevertheless, the ROC curves of classifiers consisting of more than 100 weak classifiers resemble each other closely for very small FPRs. Since a classifier's FPR is inversely proportional to its precision, we focus our further evaluation on those small FPRs to ensure a preferably high precision which is mandatory for a proper functioning of our monitoring approach. Although the general performance of the largest cascade exceeds that of all others, the classifier consisting of 152 weak classifiers yields the lowest FPR up to a TPR of 96.2%.

The findings in Figure 6c are also supported by the results in Figure 7. The evaluation up to 150 weak classifiers shows that, in general, the accuracy of the classifier increases. If the classifier contains 152 weak classifiers, it reaches an accuracy of 98 % on the validation set. Above 150 weak classifiers, the accuracy of the classifier decreases slightly on the validation set. In contrast, the accuracy increases slightly on the training set. This gives rise to suggesting a slight overfitting.

Based on these findings, we conclusively train a soft cascaded classifier for the application in our approach to the monitoring of workers on construction sites. We provide a pool of feature descriptors consisting of Haar-like features and five-bin color histograms. We add feature descriptors beginning with a size corresponding to 10% of the samples' sizes and iteratively increase their sizes by further 10% up to a size of 100% of the samples' sizes. For training, we initially provide 800 samples, each positive and negative, randomly chosen from our training dataset. We then let AdaBoost choose 152 weak classifiers during training which classify the set of training samples best. This halves the time required for the later classification process while retaining an acceptable detection rate. After calibrating this soft cascaded classifier and integrating it into our classical detection setup, we are able to process images at about 28 fps, which even exceeds the frame rate of the camera used for our experiments.



(**a**) Feature size

(**b**) Number of samples

(**c**) Number of weak classifiers

Figure 6. ROC curves indicating the effect of different hyperparameters on the detection quality.

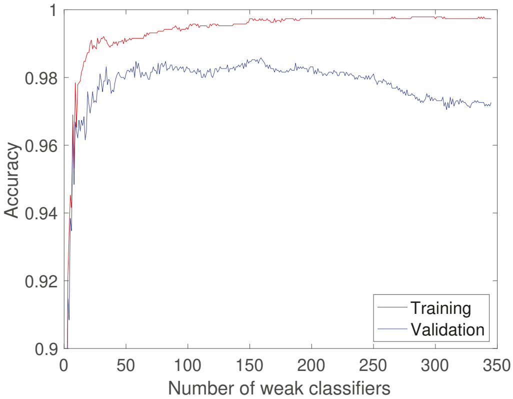


Figure 7. Calculated accuracy of the classifier when varying the number of weak classifiers.

3.2. Optimization of YOLOv3

Prior experiments showed that YOLO's detection results are considerably robust concerning changes in hyperparameters such as the learning rate, momentum or weight decay. According to these experiments, an optimization of these parameters would not significantly improve the detection results. Therefore, we use these already empirically collected values in our work and do not perform any fine tuning. The most important factor, however, is the size of the input image, since this influences both the detection speed and the precision of the detector. Thus, we conduct a grid search to find the optimal input image size for our purpose. For this purpose, we measure the detection speed in milliseconds (ms) for each image size and calculate the mean Average Precision (mAP) to determine the detection accuracy. Due to YOLO's downsampling architecture the input image size should be a multiple of 32 px in width and height. We begin the grid search at a factor of 13 for both image dimensions resulting in an image size of 416 × 416 px and gradually increase the factor by 3 up to a final input image size of 800 × 800 px. All these instances of YOLO are trained on the entire training dataset described in Section 2.1 and are evaluated on the validation dataset.

We found that the detection accuracy generally rises with an increasing image size, although the detection speed drops simultaneously (see Table 1). At the maximum image size of 800 × 800 px, YOLO yields the best detection accuracy. The processing time for a single image, however, is 97 ms which corresponds to about 10 fps when processing each image of a video. With a processing time of 46 ms per image (22 fps), using the minimal input image size is still too slow to run the detection on every single frame of our video sequences in real-time. Since our camera captures frames at 25 fps, we decided to use an input image size of 608×608 px as tradeoff. This way, we are able to process at least every second frame with a desirable precision.

Table 1. The results of detection accuracy and detection speed for different image sizes.

Image Size (px)	Detection Speed (ms)	mAP
416 × 416	46	72.9
512 × 512	56	75.2
608 × 608	67	78.5
704 × 704	83	80.3
800 × 800	97	82.7

4. Results

In our experiments, we examined the detection quality as well as the capabilities of the approaches, as the basis of a tracking system. To determine their detection quality, we applied both optimized approaches to our test data (see Section 4.1). Afterwards, we used both detection methods for tracker initialization and the recognition of already tracked workers. We examined the precision of the resulting tracks as well as the general ability to accurately identify construction workers, as shown in Section 4.2.

4.1. Detection Quality

To contrast the performances of the optimized detection methods introduced in this paper, we applied both methods to the evaluation dataset described in Section 2.1. For the comparison, we adduced the particular precision and recall of each method on this dataset. Since tracking the workers in a centimeter-perfect manner is usually not required, we defined an Intersection over Union (IoU) value of at least 0.6 to be sufficient to indicate a true positive detection. On our evaluation data, the classical approach exhibits a recall of 96.2% with 99.8% precision. Contrarily, YOLO only achieves a recall of 88.2% with a similar precision of 99.2% using a confidence threshold of 0.9. Even reducing the threshold to 0.5 results in a recall of only 93.5% while precision drops to 97.0%.

4.2. Tracking

Keeping track of the workers is the main purpose of monitoring applications. Accordingly, the tracking should be preferably accurate to provide satisfying results. The detectors of such monitoring systems are the key components for a reliable tracking as their detections serve as initializations and updates for the tracks. In this experiment, we compared the performances of both detectors developed in this paper with respect to the aforementioned requirements. We applied Kalman filtering to their detections to implement a simple and easy to evaluate tracking system.

Each generated tracker has the same dimension during evaluation. The selected size was set manually before, so that the rectangle completely encloses the construction worker in the center of the image. If the detector does not detect a construction worker who is already tracked, the Kalman filter determines its next position only using the prediction. This allows tracking a construction worker who is in motion but is covered, for example, by an object temporarily. Since the accuracy of the position determination decreases without correction of the motion model, these predictions are executed up to a maximum of one second. Otherwise, the track is erased and a new one is set up for the construction worker when the detector detects him again. The prediction is also used to mark the direction of the worker's movement in the image. For this, the next position in the current image is predicted. Construction workers who move towards a dangerous area can be better identified this way.

The precision and continuity of the underlying detector primarily determines the accuracy and robustness of the tracks. We relied on the approach of Xiao and Zhu [36] to measure these metrics. They proposed the average sequence overlap score (AOS) and center location error ratio (CER) measures for accuracy and track length (TL) for robustness. Ambiguity errors caused by tracks crossing each other are not taken into account since this is a feature of the tracker rather than of the detector. We adapt those metrics to fit to our experimental setup, as shown in Equations (2)–(4).

$$\mathrm{AOS} = \frac{1}{n}\sum_{t=1}^{n}\frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}, \tag{2}$$

$$\mathrm{CER} = \frac{1}{n}\sum_{t=1}^{n}\frac{||C_t^G - C_t^T||_2}{\mathrm{size}(A_t^G)}, \tag{3}$$

$$\mathrm{TL} = \frac{n}{N}, \tag{4}$$

where t represents the current time step, n denotes the number of video frames in which a worker is tracked and N is the number of frames in which the worker is present. The worker's bounding box areas are indicated by A and their centers by C where the superscripts G and T mark ground truth and tracked bounding boxes, respectively. Finally, $|| \circ ||_2$ denotes the two dimensional euclidean distance and $\mathrm{size}(\circ)$ represents the two dimensional size of an area.

For an eligible comparison of their performances, we evaluated the tracking systems, emerging from each detector combined with Kalman filtering, to identical scenes of construction sites. These scenes are chosen as described in Section 2.1. None of them have been shown to any detector before, neither during training nor during calibration and validation. In the following, we refer to Scene 1 as the scene of four pedestrian workers from the video sequence which is well illuminated, while Scene 2 denotes its overexposed counterpart from the other sequence. The scene showing a car approaching a worker is referred to as Scene 3. The tracking results using the classical detection approach are given in Section 4.2.1. Section 4.2.2 discusses the results of the tracker relying on YOLOv3.

4.2.1. Tracking Results Using a Classical Machine Learning Detector

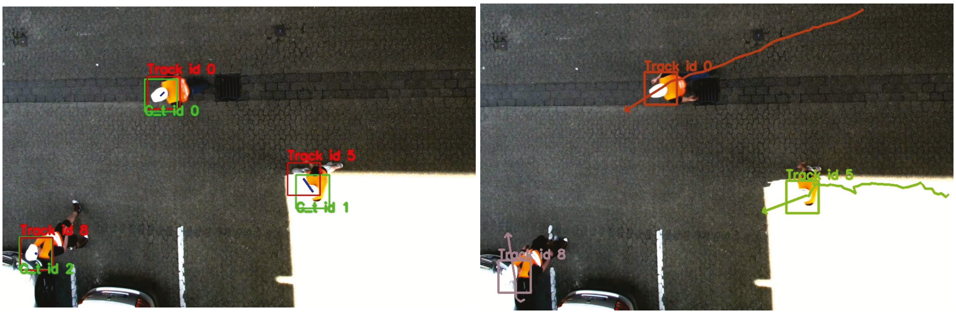
We applied the tracking system based on the classical machine learning detector to all three evaluation scenes. To each resulting track, a random ID was assigned. Further, we determined the AOS, CER and TL for each track separately. The performance of this system on each of the three evaluation scenes is summarized in Table 2.

Table 2. Results of the classical machine learning tracking system applied to all three evaluation scenes. For each scene the resulting measures according to Equations (2)–(4) are given. Tracking IDs are assigned randomly to each particular track. The last row highlights the performance of this system averaged over all scenes.

	ID	AOS	CER	TL
Scene 1	0	0.92	0.002	0.97
	5	0.65	0.004	0.92
	8	0.90	0.002	0.99
	9	0.87	0.002	1
Scene 2	1	0.69	0.004	0.92
	2	0.88	0.002	0.95
	3	0.79	0.003	0.97
	4	0.88	0.002	0.88
Scene 3	0	0.82	0.004	1
Average		0.82	0.003	0.96

The results indicate a significant decrease in tracking quality for overexposed scenes. This becomes clear when comparing the results of Track 5 with those of the other tracks of the first scene. As shown in Figure 8, Track 5 is located in a highly overexposed section of the scene, whereas the other tracks traverse well-illuminated sections only. This complicates the detection and the tracking consequently results in an inaccurate location, as illustrated in Figure 8a. This figure shows the AOS measure by comparing ground truth data in green to the actual tracks in red and the CER measure by blue lines indicating the distance between the labels' centers. As can be seen, Tracks 0 and 8 in the well-illuminated area match the ground truth closely and their centers are also close to each other. In contrast, the overlapping area of Track 5 in the highly overexposed section and its corresponding ground truth label is substantially smaller, whereas their centers' distance is considerably large. Such deviations during the tracking impair its accuracy which becomes visible from the jagged walking path determined for Track 5 in Figure 8b. This is also supported by the TL of only 0.92. Workers in overexposed areas are harder to detect so that the tracking begins delayed, which results in a shortened track length.

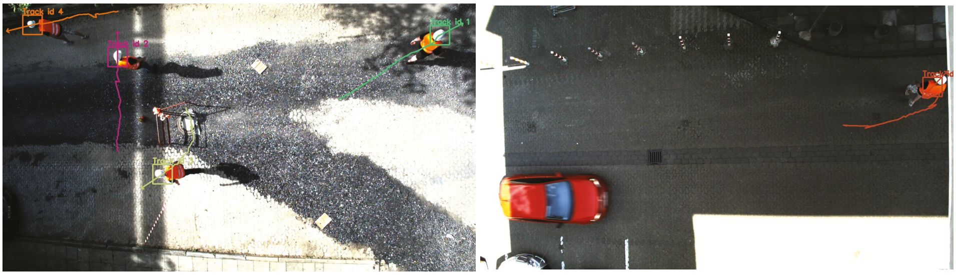
In Scene 2, these effects are much less pronounced since the scene is only slightly overexposed. As the comparison of the AOS shows, the difference in the labels' overlap of Scenes 1 and 2 is only 0.025 on average, and, even if the outlier (Track 5) in Scene 1 is disregarded, the difference raises to only about 0.086. The average TL in this scene also decreases only moderately compared to Scene 1. By depicting an example of Scene 2, Figure 9a illustrates that the quality of our tracking approach is sufficient even on slightly overexposed scenes. This finding is supported by measuring the TL and the CER. Both do not vary significantly from Scene 1 to Scene 2. All construction workers are consistently tracked almost throughout the entire duration of both scenes. By applying the tracking system to Scene 3, we show its behavior in the case of moving objects which are not pedestrian construction workers. The results for Scene 3 in Table 2 depict that only one track is set up. As graphically confirmed by Figure 9b, this track belongs to the only worker in this scene. The worker is tracked successfully and no further tracks for the non-worker objects are mistakenly generated. Again, the AOS achieved by the classical system is within an acceptable range, and CER and TL confirm that the worker is tracked consistently throughout the whole scene's duration.



(**a**)

(**b**)

Figure 8. Example of the tracking performance of the classical approach on Scene 1. (**a**) Illustration of the AOS and CER measures. Ground truth labels (green) are compared to the actual tracks (red). Blue lines indicate the distance between the labels' centers. (**b**) Tracking result. The worker's current positions are given by randomly colored squares. Equally colored lines illustrate their previous walking paths. Arrows show their predicted walking direction.



(**a**) (**b**)

Figure 9. Examples of the classical tracking system applied to different scenes. (**a**) An example of the tracking system applied to Scene 2 shows that the tracking is sufficient even on slightly overexposed scenes. (**b**) On Scene 3, the tracking results show that the worker is tracked successfully while the moving vehicle remains untracked as intended.

4.2.2. Tracking Results Using a Deep Learning Detector

To ensure a proper comparison, the evaluation of the deep learning-based tracking system is done analogously to the evaluation of the classical approach. Again, IDs are assigned randomly to the tracks, and the accuracy and robustness metrics are determined per track. Table 3 summarizes the performance results of the tracking.

Table 3. Results of the deep learning-based tracking system applied to all three evaluation scenes. For each scene, the resulting measures according to Equations (2)–(4) are given. Tracking IDs are assigned randomly to each particular track. The last row highlights the performance of this system averaged over all scenes.

	ID	AOS	CER	TL
Scene 1	0	0.94	0.002	0.98
	1	0.87	0.003	0.98
	2	0.96	0.002	1
	5	0.98	0.002	0.95
Scene 2	0	0.96	0.001	0.99
	1	0.95	0.005	1
	2	0.98	0.002	0.99
	3	0.90	0.004	1
Scene 3	0	0.98	0.002	0.98
Average		0.95	0.003	0.98

The tracking system exhibits a satisfactory performance on average over all sequences. Especially on well-illuminated scenes, a high accuracy is provided as indicated by the AOS and CER of Scene 1. The system's performance on this scene is very robust since all workers were tracked almost over the entire length of this video sequence. However, Track 1, which is located in a heavily overexposed area, shows that overexposure affects the quality of the tracking. As the TL shows, the track remains robust, but the accuracy measured by the AOS decreases by about 10% This is also confirmed by a slight increase in the CER. Nevertheless, the system performs very well if the scene is only slightly overexposed as is the case in Scene 2. Here, the AOS barely indicates any negative effects despite the overexposure. Only the CER shows a minor increase. Similar to the highly overexposed case, the TL measurements again emphasize the tracking system's robustness, although slight overexposure is present on the entire scene.

Besides this, the results of this experiment also show that the system does not confuse construction workers with any other object in the scenes. As Table 3 shows, the correct number of tracks was set for each scene. Alongside the four construction workers, Scenes 1 and 2 include static construction related objects, such as traffic cones and barriers. These stay correctly undetected and untracked during the course of the video sequences. Moving objects such as the car in Scene 3 also remain correctly unnoticed by the tracker.

5. Discussion

As shown by our experiment regarding the detection quality, both methods outperform the approach of Luo et al. [24] by far. While Luo et al. achieved a precision of 75.1%, our approaches reach 99.8% and 99.2%, respectively. A proper comparison of the results is delicate since the datasets used for training and evaluation differ from each other. However, tests using YOLO with the same input image size as proposed by Luo et al. yield similar results, which indicates at least a weak comparability of the setups. This confirms that the input image size is a crucial parameter for a proper detection when using YOLO, as the objects to be detected shrink proportionally to the size of the images. Nevertheless, this experiment also shows that the results of the classical method exceed those of YOLO by about 5%. As mentioned above, the image size may be a reason for this. The classical approach operates on the original high resolution, exhibiting more detailed features than the drastically downscaled images used by YOLO. A second reason may arise from the limited dataset since it is known that CNNs require a vast amount of training data. The number of provided training samples may have been too small to sufficiently adapt to the class of workers ,although we deliberately used a pre-trained network which already developed general feature descriptors for classification. On the contrary, the findings in Section 3.1 highlight that classical computer vision approaches cope significantly better with fewer samples than CNNs.

Both reasons emphasize common practical issues concerning data gathered in the context of construction site monitoring. Since computer vision approaches are rarely applied in the field, dataset generation has rarely been regarded a subject until now. Beyond this, generating a reasonable dataset covering all environmental and lighting conditions is extremely time-consuming and tedious. Given such a dataset, the small size of the pedestrian workers within the images remains to be a limiting factor. Thus, choosing a classical detection approach over a CNN is desirable to obtain the best detection results for the purpose of monitoring pedestrian workers.

However, our tracking experiments showed that both approaches yield suitable results. As can be seen from the results of Scene 1, the evaluated systems perform similarly well on a well-illuminated environment. Both approaches exhibit an excellent accuracy with very low CER and high AOS of about 90%. Furthermore, the high TL rates indicate their robustness. The slight deviations from the optimum are mainly due to workers entering and leaving the scene. In these situations, workers are located at the image boarders and may be only partly visible. This complicates the detection of the workers so that tracks may be set up late or may terminate early. Apart from these effects, the tracking performed by both approaches is very precise despite the CNN's limitations in detection quality and speed. This shows that even simple tracking methods can compensate for lower detection rates. However, Scene 1 also shows that both approaches have issues with high overexposure. In both cases, the AOS of the regarding track drops noticeably accompanied by a raise in CER. The worker is still tracked sufficiently by both systems but the accuracy dramatically suffers from the overexposure. The CNN-based approach seems to cope with this issue slightly better than the classical system. This is no reliable statement yet as there is only a single instance of evidence available in the data. The results of Scene 2 provide further insights into the subject of overexposure. On this scene, the TL rates remain almost stable indicating a satisfying robustness. This ensures reliable tracking with both tracking systems despite a certain degree of overexposure. However, the CNN-based approach achieves significantly better AOS rates in this slightly overexposed environment. The CER rates fluctuate more than with the classical approach. On the one hand, the considerable decrease in AOS shows that the classical system has difficulties in identifying the worker's dimensions precisely when overexposed. This is also supported by the findings regarding the overexposed Track 5 of Scene 1. At the same time, the stable CER rates indicate that the worker's centers—and thereby their locations—are recognized with high accuracy. On the other hand, the CNN-based system reveals its weakness concerning a precise localization, as shown by the varying CER. Instead, it accurately determines a worker's dimensions despite aggravated conditions. With this knowledge, the size of the detected area could in the future be passed to the tracker so that the worker is for the most part completely marked on the image. For the classical approach, a fixed size of the tracking box would be more suitable, which has to be determined in advance depending on the camera height.

This points out that both approaches possess certain pros and cons. Under ideal conditions both perform similarly well but as conditions change, they start focusing on different aspects. While the classical approach precisely keeps track of the worker's locations, the CNN-based system accurately recognizes their dimensions. Accordingly, a conclusive decision has to be made with respect to the demands of particular applications.

6. Conclusions

In this study, we investigated whether deep learning methods surpass classical approaches on construction worker monitoring despite their limitations. We chose YOLOv3 for the CNN and a classical approach based on a soft cascaded classifier as representatives for our comparison. The trained detectors were then embedded in a tracking system to track construction workers in video sequences. To evaluate the tracking systems under various conditions, we generated different video sequences. These contain different environmental and lighting conditions as well as stationary and moving non-worker objects. Both tracking systems were applied to the same sequences to ensure a proper comparison.

As our experiments showed, the classical approach clearly outperforms the CNN on the detection task in terms of quality and speed. The lack of quality is most likely due to an insufficient amount of training data and the heavy downscaling of the images. The low detection speed of substantially less than 22 fps is affiliated to the high computational costs. However, the tracking experiment showed that the CNN's drawbacks are fully compensated even by a simple tracking method. Both approaches showed satisfying results when tracking workers under ideal conditions. They were even able to suppress a false tracking of any stationary or moving non-worker object. Nevertheless, both tracking systems reveal deficiencies when applied to overexposed conditions. While the CNN keeps precise track of the worker's dimensions, localization becomes inaccurate. The classical approach behaves exactly vice versa. According to these findings, there is no optimal monitoring approach in general. A suitable approach has to be chosen with respect to the demands of the particular application.

With our example scenarios, we showed that with both approaches a satisfactory tracking of the construction workers from the top view is possible. Nevertheless, it would be an advantage to know whether satisfactory tracking is also achieved with more complex work processes with different moving machines, different construction site constellations and different weather conditions. Therefore, future work should compare the presented tracking approaches with scenes that have a different focus. In this case, optical flow techniques can additionally be used to enable tracking also with cameras mounted, e.g., on the crane boom. The background subtraction would consider known crane movements and thus dynamically changing backgrounds would be taken into account. Furthermore, future work should concentrate on substantially increasing the size of the datasets. By this, the training of CNN-based approaches can be improved. This helps the detector to develop a better generalization ability, which may advance the tracking results especially under difficult conditions. Reasonable strategies for the extension may be alternative data augmentation techniques as well as computer generated data.

Author Contributions: Conceptualization, M.N.; data curation, M.N.; investigation, M.N.; methodology, M.N.; project administration, M.N.; software, M.N. and D.P.; supervision, M.K.; validation, D.P.; visualization, D.P.; writing—original draft, M.N.; and writing—review and editing, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sadeghpour, F.; Teizer, J. Modeling Three-Dimensional Space Requirements for Safe Operation of Heavy Construction Equipment. In Proceedings of the Fifth International Conference on Construction in the 21st Century: Collaboration and Integration in Engineering, Management and Technology, Istanbul, Turkey, 20–22 May 2009; pp. 740–747.
2. Kazan, E.; Usmen, M.A. Worker safety and injury severity analysis of earthmoving equipment accidents. *J. Saf. Res.* **2018**, *65*, 73–81. [CrossRef] [PubMed]
3. OSHA Training Institute. *Construction Focus Four: Outreach Training Packet*; Technical Report; OSHA Directorate of Training and Education: Washington, DC, USA; 2011.
4. Tabassi, A.A.; Ramli, M.; Bakar, A.H.A. Training and Developement of Workforces in Construction Industry. In *Emerging Issues in the Natural and Applied Sciences*; Progress IPS LLC.: Baku, Azerbaijan, 2011.
5. Teizer, J.; Vela, P. Personnel tracking on construction sites using video cameras. *Adv. Eng. Inform.* **2009**, *23*, 452–462. [CrossRef]
6. Nasr, E.; Shehab, T.; Vlad, A. Tracking Systems in Construction: Applications and Comparisons. In Proceedings of the Annual Conference of Associated Schools of Construction, San Luis Obispo, CA, USA, 10–13 April 2013.
7. Carbonari, A.; Giretti, A.; Naticchia, B. A Proactive System for Real-Time Safety Management in Construction Sites. *Autom. Constr.* **2011**, *20*, 686–698. [CrossRef]
8. Park, M.W.; Brilakis, I. Construction Worker Detection in Video Frames for Initializing Vision Trackers. *Autom. Constr.* **2012**, *28*, 15–25. [CrossRef]
9. Khoury, H.M.; Kamat, V.R. High-Precision Identification of Contextual Information in Location-Aware Engineering Applications. *Adv. Eng. Inform.* **2009**, *23*, 483–496. [CrossRef]
10. Behzadan, A.H.; Kamat, V.R. Georeferenced Registration of Construction Graphics in Mobile Outdoor Augmented Reality. *J. Comput. Civ. Eng.* **2007**, *21*, 247–258. [CrossRef]
11. Xie, P.; Petovello, M.G. Measuring GNSS Multipath Distributions in Urban Canyon Environments. *IEEE Trans. Instrum. Meas.* **2015**, *64*, 366–377. [CrossRef]
12. Jang, W.S.; Skibniewski, M.J. Embedded System for Construction Asset Tracking Combining Radio and Ultrasound Signals. *J. Comput. Civ. Eng.* **2009**, *23*, 221–229. [CrossRef]
13. Chae, S.; Yoshida, T. Application of RFID Technology to Prevention of Collision Accident with Heavy Equipment. *Autom. Constr.* **2010**, *19*, 368–374. [CrossRef]
14. Lu, W.; Huang, G.Q.; Li, H. Scenarios for Applying RFID Technology in Construction Project Management. *Autom. Constr.* **2011**, *20*, 101–106. [CrossRef]
15. Skibniewski, M.J.; Jang, W.S. Simulation of Accuracy Performance for Wireless Sensor-Based Construction Asset Tracking. *Comput.-Aided Civ. Infrastruct. Eng.* **2009**, *24*, 335–345. [CrossRef]
16. Pradhan, A.; Ergen, E.; Akinci, B. Technological Assessment of Radio Frequency Identification Technology for Indoor Localization. *J. Comput. Civ. Eng.* **2009**, *23*, 230–238. [CrossRef]
17. Juels, A. RFID Security and Privacy: A Research Survey. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 381–394. [CrossRef]
18. Seo, J.; Han, S.; Lee, S.; Kim, H. Computer Vision Techniques for Construction Safety and Health Monitoring. *Adv. Eng. Inform.* **2015**, *29*, 239–251. [CrossRef]

19. Chi, S.; Caladas, C.H. Automated Object Identification Using Optical Video Cameras on Construction Sites. *Comput.-Aided Civ. Infrastruct. Eng.* **2010**, *26*, 368–380. [CrossRef]
20. Memarzadeh, M.; Golparvar-Fard, M.; Niebles, J.C. Automated 2D detection of construction equipment and workers from site video streams using histograms of oriented gradients and colors. *Autom. Constr.* **2013**, *32*, 24–37. [CrossRef]
21. Yang, J.; Arif, O.; Vela, P.A.; Teizer, J.; Shi, Z. Tracking multiple workers on construction sites using video cameras. *Adv. Eng. Inform.* **2010**, *24*, 428–434. [CrossRef]
22. Luo, X.; Li, H.; Yang, X.; Yu, Y.; Cao, D. Capturing and Understanding Workers' Activities in Far-Field Surveillance Videos with Deep Action Recognition and Bayesian Nonparametric Learning. *Comput.-Aided Civ. Infrastruct. Eng.* **2018**, *34*, 333–351. [CrossRef]
23. Son, H.; Choi, H.; Seong, H.; Kim, C. Detection of construction workers under varying poses and changing background in image sequences via very deep residual networks. *Autom. Constr.* **2019**, *99*, 27–38. [CrossRef]
24. Luo, X.; Li, H.; Wang, H.; Wu, Z.; Dai, F.; Cao, D. Vision-Based Detection and Visualization of Dynamic Workspaces. *Autom. Constr.* **2019**, *104*, 1–13. [CrossRef]
25. Vierling, A.; Sutjaritvorakul, T.; Berns, K. Crane Safety System with Monocular and Controlled Zoom Cameras. In Proceedings of the International Symposium on Automation and Robotics in Construction, Berlin, Germany, 20–25 July 2018. [CrossRef]
26. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
27. Neuhausen, M.; Teizer, J.; König, M. Construction Worker Detection and Tracking in Bird's-Eye View Camera Images. In Proceedings of the International Symposium on Automation and Robotics in Construction, Berlin, Germany, 20–25 July 2018; pp. 1159–1166. [CrossRef]
28. BlockCorp. X2 Crane Camera System. 2020. Available online: https://www.blokcorp.com/products/blokcam/x2-crane-camera-system/ (accessed on 14 November 2020).
29. Zivkovic, Z. Improved Adaptive Gaussian Mixture Model for Background Subtraction. In Proceedings of the 17th IEEE International Conference on Pattern Recognition, Cambridge, UK, 26 August 2004; pp. 28–31. [CrossRef]
30. Kim, H.; Kim, K.; Kim, H. Vision-Based Object-Centric Safety Assessment Using Fuzzy Inference: Monitoring Struck-by Accidents with Moving Objects. *J. Comput. Civ. Eng.* **2016**, *30*, 04015075. [CrossRef]
31. Suzuki, S.; Abe, K. Topological structural analysis of digitized binary images by border following. *Comput. Vis. Graph. Image Process.* **1985**, *30*, 32–46. [CrossRef]
32. Dollár, P.; Tu, Z.; Perona, P.; Belongie, S. Integral Channel Features. In *Proceedings of the British Machine Vision Conference*; BMVC Press: London, UK, 2009; pp. 91.1–91.11. [CrossRef]
33. Bourdev, L.; Brandt, J. Robust Object Detection via Soft Cascade. In Proceedings of the Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2005; Volume 2, pp. 236–243. [CrossRef]
34. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
35. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
36. Xiao, B.; Zhu, Z. Two-Dimensional Visual Tracking in Construction Scenarios: A Comparative Study. *J. Comput. Civ. Eng.* **2018**, *32*, 04018006. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Using Synthetic Data to Improve and Evaluate the Tracking Performance of Construction Workers on Site

Marcel Neuhausen *, Patrick Herbers and Markus König

Chair of Computing in Engineering, Ruhr-University Bochum, 44801 Bochum, NRW, Germany; patrick.herbers@ruhr-uni-bochum.de (P.H.); koenig@inf.bi.ruhr-uni-bochum.de (M.K.)
* Correspondence: marcel.neuhausen@ruhr-uni-bochum.de

Received: 19 March 2020; Accepted: 16 July 2020; Published: 18 July 2020

Abstract: Vision-based tracking systems enable the optimization of the productivity and safety management on construction sites by monitoring the workers' movements. However, training and evaluation of such a system requires a vast amount of data. Sufficient datasets rarely exist for this purpose. We investigate the use of synthetic data to overcome this issue. Using 3D computer graphics software, we model virtual construction site scenarios. These are rendered for the use as a synthetic dataset which augments a self-recorded real world dataset. Our approach is verified by means of a tracking system. For this, we train a YOLOv3 detector identifying pedestrian workers. Kalman filtering is applied to the detections to track them over consecutive video frames. First, the detector's performance is examined when using synthetic data of various environmental conditions for training. Second, we compare the evaluation results of our tracking system on real world and synthetic scenarios. With an increase of about 7.5 percentage points in mean average precision, our findings show that a synthetic extension is beneficial for otherwise small datasets. The similarity of synthetic and real world results allow for the conclusion that 3D scenes are an alternative to evaluate vision-based tracking systems on hazardous scenes without exposing workers to risks.

Keywords: construction productivity; construction safety; deep learning; synthetic data; tracking

1. Introduction

Vision-based detection and tracking have already found their way in a wide area of applications. Pedestrian detection has become an essential topic in the modern automotive industry and is also a relevant part of various surveillance systems. Even sports analytics make use of these techniques for the assistance of referees as well as for the automated generation of game statistics. Despite the huge potential, such approaches are rarely employed in the construction sector nowadays. Especially processes on construction sites could benefit from computer vision methods. As construction sites are complex and continuously changing environments, keeping track of the ongoing processes can be challenging. Various trades are involved at each stage of construction and different work orders are executed by workers and construction machines simultaneously. On the one hand, this can affect an individual worker's workflow since the complex processes are hard to grasp. On the other hand, workers can easily lose track of the ongoing processes in their surroundings while concentrating on their own tasks. This inadvertency may result in hazardous situations. Excavator drivers may fail to notice unaware pedestrian workers crossing their paths. Also, crane operators may lift their loads over workers standing in blind spots.

By monitoring pedestrian workers on site, their current workflows can be optimized [1]. Furthermore, assistance systems can support machine operators in avoiding hazards involving workers [2]. However, a reliable and precise tracking system is a prerequisite for both applications. A multitude of approaches in this field apply tag-based methods where tags are attached to all objects to be monitored. Depending on the specific use case, Radio-Frequency Identification (RFID), Ultra-Wideband (UWB), or Global Navigation Satellite System (GNSS) technology is employed [3]. Approaches using GNSS are usually applied to the localization of equipment, workers, and machines in spacious outdoor construction environments [4,5]. Near to large structures like buildings, walls, and other construction components, GNSS becomes unreliable as it suffers from multipath effects [6]. Tracking workers in the proximity of heavy construction machinery or cranes carrying large loads may, thus, become too inaccurate for operator assistance systems. Accordingly, RFID- and UWB-based approaches are commonly used to improve the safety during construction. By equipping workers with RFID tags, operators get warned if a worker enters the range of a machine's tag reader [7]. Warning workers of entering hazardous areas can be achieved by positioning the workers using UWB tags and readers [8]. Nevertheless, in general a precise localization by means of radio-frequency technologies remains challenging [9]. Beyond that, all tag-based approaches involve high costs for the amount of required tags and readers [3]. Additionally, wearing such tags causes discomfort for the workers [10]. Camera-based monitoring approaches overcome these deficiencies. They allow for the tracking of workers even close to large structures as they neither suffer from multipath effects nor from other signal interferences leading to an inaccurate positioning. Additionally, they constitute an affordable alternative to tag-based solutions since only a few cameras are required to monitor large areas of construction sites.

In recent years, some effort has already been made to monitor pedestrian workers using computer vision techniques. Firstly, pose estimation is used to recognize specific sequences of movement as well as unsafe actions [11,12]. Secondly, workers are detected in video images using Support Vector Machine (SVM) and k-Nearest Neighbors (k-NN) classifiers [13] and are then tracked over time [14]. More recent approaches employ Convolutional Neural Networks (CNNs) for both detection and tracking purposes [15,16]. However, the detection results of such approaches are in need of improvement. Park and Brilakis [13] state recall rates of 87.1 % to 81.4 % at precision rates ranging from 90.1 % to 99.0 %, respectively. Luo et al. [15] report an Average Precision (AP) of 75.1 % with an Intersection over Union (IoU) threshold of only 0.5. Although these results are as of yet insufficient for the implementation of a reliable safety-relevant system, its considerable potential can already be inferred. To improve the detection allowing for an uninterrupted tracking, the training of the underlying machine learning algorithms has to be enhanced. Furthermore, the subsequent evaluation of those systems has to be extended in order to model a broader variety of scenarios which may occur during a real application. This ensures a reliable tracking during productive operation even under challenging conditions. Training and evaluation both require a vast amount of data. This especially holds for deep learning approaches as they usually require a multitude of training samples compared to classical machine learning techniques. However, sufficient datasets rarely exist for construction site scenarios and gathering a comprehensive dataset is time consuming and tedious. Although common data augmentation techniques, such as randomly cropping, flipping, or rotating samples, enable the extension of existing datasets, the effect to the training of a detector is limited. While this increases the variability in the dataset, adapting the training towards scenarios not contained in the initial dataset is unfeasible. Moreover, assembling realistic evaluation scenarios might also require exposing workers to hazards to determine the behavior of a tracking system in such situations.

A solution to the data problem may be synthetic data. Parker [17] defines synthetic data as "any production data applicable to a given situation that are not obtained by direct measurement". While this definition was originally meant for mathematical models, it can also be applied to machine learning datasets. The complexity of synthetic data generation can range from simple mathematical equations, to fully simulated virtual environments. Synthetic data was mainly used in software development processes, but has lately found application among machine learning research. Generating required data synthetically might be desirable as synthetic data has the ability to extend existing datasets or to create new datasets with a significant reduction in effort. While traditional datasets have to be aggregated and labeled by hand, synthetic data can be accurately labeled automatically. As the ground truth is already available in a simulated environment, labeling becomes trivial. Synthetic data can be created completely from scratch (fully synthetic data) or based on real datasets for data augmentation (partially synthetic data). For example, Jaderberg et al. [18] created a dataset for text recognition by synthetically generating distorted, noisy images of text from a ground truth dictionary. Gupta et al. [19] improved on the idea of synthetic text recognition by rendering text onto real images. Tremblay et al. [20] used synthetic data as a form of domain randomization where random objects are placed in various environments. The amount of possible combinations enables a CNN to recognize the important part of a picture with minimal effort in dataset generation. The size of a synthetic dataset based on a 3D environment is nearly unlimited as can be seen in the SYNTHIA dataset [21]. Utilizing dynamic objects and render settings, scenarios can be varied indefinitely. One popular technique is co-opting video games with realistic graphics for various imaging tasks, such as depth estimation [22] or image segmentation [23]. Furthermore, synthetic datasets can be augmented by creating variations of lighting or environmental conditions like rain or fog. Tschentscher et al. [24] created a synthetic dataset of a car park in Unreal Engine 4 where camera angles and weather conditions can be changed at will. Through the use of this dataset, Horn and Houben [25] improved a k-NN based parking space detection algorithm by six percentage points while simultaneously reducing the time for generating the dataset.

As this paper focuses on the improvement of the detection and tracking of construction workers, a few particularities have to be taken into consideration. Construction sites depict more complex and harder to model environments than the well structured car park scenarios in [25]. Sites are essentially less structured and subject to greater changes. Construction material, machinery, and workers operate or interact with each other almost everywhere on the site. In the course of this, occlusions of the workers by construction machines and their loads occur frequently. Additionally, modeling human shapes and behaviors realistically is still a challenging task. Compared to the simple trajectories of cars, a worker's movement is significantly more complex and harder to predict. Dynamic environments such as these require a diversified dataset that covers all cases of working on a construction site, including dangerous situations which would be unethical to re-enact in a real world scenario.

Lately, image style transfer learning such as Cycle-Consistent Generative Adversial Networks (CycleGANs) [26] has been used for data augmentation. CycleGANs are able to learn a bidirectional image style mapping between two unlabeled image domains. The two image sets for training the GAN do not need to be paired sets, but there is still a significant amount of training data required. Wang et al. [27] use this technique to enhance their synthetic dataset for crowd counting by augmenting the synthetic dataset with image styles from existing datasets. Since the synthetic data in [27] is collected from a video game, the image style is significantly different from actual camera footage. However, the small size and low variation in environmental conditions of common real world datasets of construction sites complicate the training of a GAN. Additionally, in this work, full control over the synthetic scenes' setup is required to sufficiently model potential hazardous situations, which includes the image style. Thus, this paper does not use CycleGANs for data augmentation.

Hence, the aim of this paper is to investigate the usage of synthetically generated data for improving both, the training and the evaluation of construction site monitoring systems. For this, several 3D scenes of construction sites with varying lighting and environmental conditions are modeled. Sequences of these scenes are extracted to extend a self-recorded real world dataset. In order to identify the effects of synthetic data, a monitoring system is built that tracks pedestrian construction workers over time. The system detects workers in consecutive video frames using YOLOv3 [28] and tracks them by Kalman filtering. To prove the concept of this paper, first the detector's performance is evaluated when it is trained on the real world dataset only. Then, the results are compared to the performance when synthetic data is gradually added to the set of training samples. We found that providing well-chosen computer generated scenes to the training set improves the detection performance by

7.5 percentage points in mean average precision (mAP). In the end, the detector achieves a mAP of 94.0 %. If inaccuracies resulting from manual labeling and image scaling are taken into account, the mAP even increases to 99.2 %. In another setting, the tracking performance of the proposed monitoring system on real world and synthetic scenarios is compared. The experiment shows that the tracking system yields similar results on both kinds of datasets. This illustrates that synthetic data provides a reasonable alternative for the evaluation of the tracking performance on hazardous situations without exposing workers to risks.

The remainder of this paper is organized as follows: Section 2 introduces the datasets as well as the tracking system. Section 3 summarizes two experiments and their results to prove the proposed concept. First, the use of synthetic data for the training of the detector is investigated in Section 3.1. Second, Section 3.2 examines the comparability of real world and synthetic data in terms of the evaluation of a tracking system. The results of these experiments are discussed in Section 4 and an exemplary use case evaluating such a monitoring system is shown. Finally, Section 5 summarizes the findings and provides an outlook to future work.

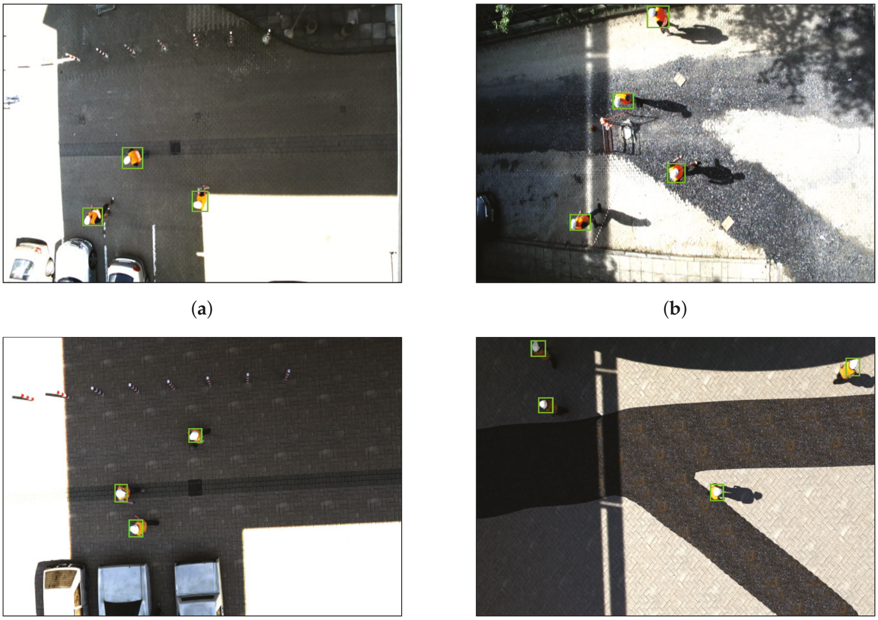
2. Materials and Methods

In order to investigate the effect of synthetically generated data on the training and evaluation of a vision-based tracking system in the scope of construction sites, we created a dataset from computer generated 3D scenes. This dataset is used to augment a small self-recorded dataset (see Section 2.1) of pedestrian workers walking across a construction site. As described in detail in Section 2.2, we modeled multiple 3D scenarios which depict various lighting and environmental conditions as well as different movement patterns for the represented workers. To evaluate our approach, we built a monitoring system that tracks pedestrian workers in video sequences. The underlying detector's hyperparameter settings are described in Section 2.3 while Section 2.4 outlines the tracking process, respectively.

2.1. Real World Dataset

Since video datasets of construction sites including pedestrian workers from a top view perspective are rarely publicly available, we assembled a small generic dataset on our own. For this, we recorded several sequences of walking construction workers using a camera mounted in a height of around 20 m. This results in a bird's-eye-like view in the center of the images whereas the view becomes oblique near the borders. A reasonable dataset should exhibit a large variation in the data samples. This includes different lighting and environmental conditions of the scenes as well as various objects and differences in their appearance. To realize variation in our dataset, we chose two environmental scenarios. These contain different ground textures and lighting conditions as well as different types of construction sites and present objects. While the ground in one scene is uniformly paved (see Figure 1a), the other scene exhibits sandy, paved, and graveled parts, as shown in Figure 1b. The scenes are recorded at different daytimes to include different but realistic lighting conditions. This results in a well-exposed to overexposed illumination in the one scene at noon whereas in the other scene, in the evening, long shadows and lateral illumination are present. The scenes also vary in their setup and item selection. One shows a roadwork-like setup containing parked and driving cars as well as striped delineators, poles, and garbage cans. The other is a building work-like scene including a construction barrier and barrier tape. In both scenes, there are areas of different illumination conditions ranging from slightly underexposed to a high overexposure. The pedestrian workers wear safety vests and helmets throughout and walk randomly across both scenes including sudden stops and directional changes. They also interact with and relocate static objects like pylons and barriers.

The resulting dataset is labeled manually to provide ground truth for training and evaluation purposes. For this, workers are identified by rectangular bounding boxes closely framing their heads, shoulders, and safety vests. In Figure 1, these labels are represented by green rectangles.



(**a**) (**b**)

(**c**) (**d**)

Figure 1. Comparing the different environmental scenarios of the real world dataset in (**a**,**b**) to their 3D generated counterparts in (**c**,**d**), respectively. Ground truth labels are marked as green rectangles.

The dataset is split into two separate subsets. A share of the recorded video sequences is assigned to each subset. This way, the construction worker detector can be trained on one subset while the evaluation is carried out on the other subset unknown to the detector until then. The shares are assigned in a way that both environmental scenarios are equally distributed within each. Each subset only obtains every tenth frame of the assigned sequences in order to ensure a sufficient amount of variation among the images. The number of workers visible in each frame varies as workers occasionally entered and left the camera's field of view during recording. Although we do not identify particular workers, the number of visible workers per image and their locations are known since the sequences were manually labeled beforehand. In the following, each visible worker per image is referred to as a worker instance. In conclusion, the subsets are organized as follows: the training set consists of 660 images of both environmental conditions containing 1000 pedestrian worker instances while the evaluation set shows 2750 worker instances in 1300 images of different sections of the recorded sequences.

2.2. Synthetic Data Generation

The synthetic dataset was built to emulate the environment and technical specifications of the real world dataset. Figure 2 summarizes the synthetic data creation process. Blender 2.80 was used for the creation of the different scenes, which were rendered using the Cycles rendering engine. All scenes were built by hand to represent real construction sites, depicting a varied set of scenes. Using a physically-based renderer allows for more realistic images than game engines, but also increases render time. For the comparison of real and synthetic data, the two real scenes were recreated (see Figure 1c,d). Overall, 8 different scenes with a total of 3835 frames and 32 tracked subjects were created for this work. Table 1 lists all created scenes, with sample frames shown in Figure 3. Similar to the real world dataset, the synthetic dataset incorporates different lighting and weather conditions, ground surface types, and surrounding clutter. As for the real dataset, this increases the variation of the data samples as well as modeling the most common conditions and construction site types.

The lighting conditions include sunny, cloudy, and rainy. The incidence angle of the sun varies to simulate the different lighting conditions at different times of the day, including long shadows in the morning and evening. Overexposure emulates the effect present in the real world dataset (compare Figure 1a,c). The virtual workers wear different safety vests (in either green, yellow, or orange) and different helmets (in either white or yellow) to ensure variability in the dataset. To ensure that the pose and movement of the virtual workers are realistic, motion capture data from the Carnegie Mellon University Motion Capture Database was used. The motion capture data was rigged onto the construction worker model to create realistic animations. Virtual workers conduct multiple different actions in the created scenes, including walking, picking up objects, interacting with each other, or repairing or using equipment. Several other moving objects are added, such as cars, forklifts, and construction material. Dangerous situations are also included, where construction material is lifted above workers' heads (see Figure 3a,d). Workers may also be partly occluded by objects.

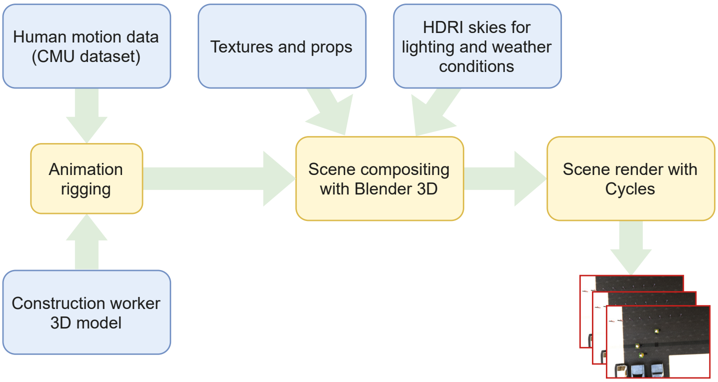

Human motion data (CMU dataset)
Textures and props
HDRI skies for lighting and weather conditions
Animation rigging
Scene compositing with Blender 3D
Scene render with Cycles
Construction worker 3D model

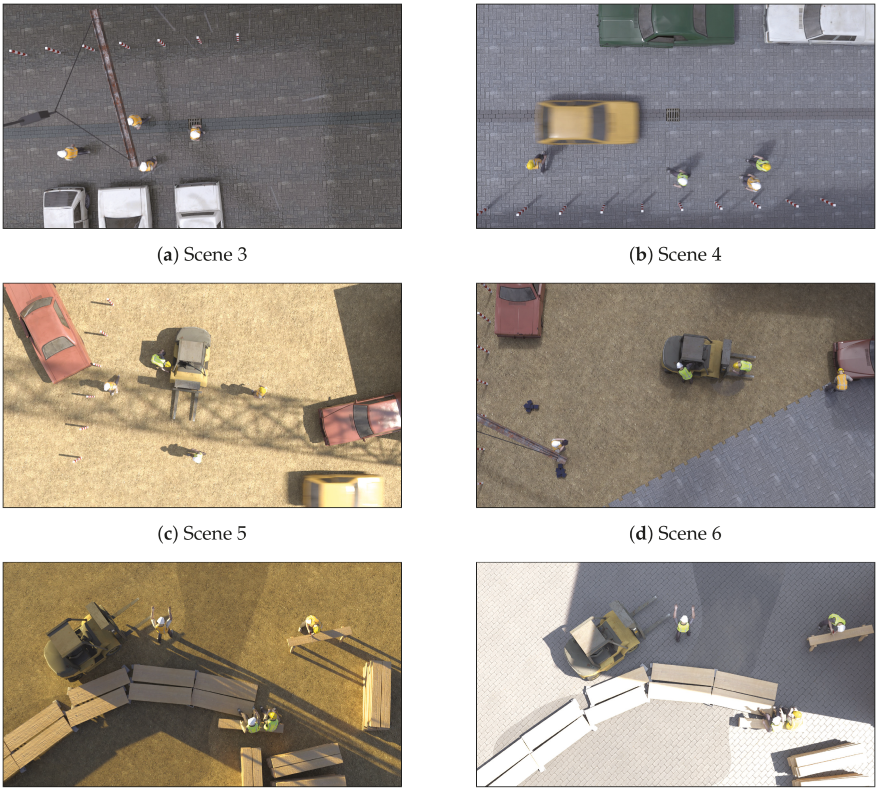
Figure 2. Process for creating synthetic scenes.

Table 1. Descriptions of the created computer generated scenes.

Scene	Description	Weather	Workers	Frames
1	Recreation of real scene 1	Sunny, partly overexposed	4	250
2	Recreation of real scene 2	Sunny	4	250
3	Same as scene 1, but in rainy conditions, and a steel beam is lifted across the workers by a crane	Rainy	4	250
4	Street, workers walk in a straight line, a car passes	Cloudy	4	500
5	Construction site, one worker working on a forklift, two workers walk past, a car drives past, one worker walks in circles	Sunny	4	600
6	Construction site, two workers working on a forklift, one worker working on a car, one worker picking up items and walking around	Cloudy	4	661
7	Construction site, one worker directs a driving forklift, two workers sit down to chat, one worker saws	Sunny, late evening	4	662
8	Same as scene 7, but with a different ground type and lighting	Sunny, slightly overexposed	4	662

Ground truth labels were extracted automatically from the 3D data. For this, a spherical hull was placed, which encompasses the head and part of the shoulders of a virtual construction worker.

The points of the hull for each subject in each frame were transformed into image coordinates, with the bounding box of the image coordinates then serving as the rectangular label. The labeling process was done completely automatically from the source 3D data, not requiring any manual labeling. Resulting labels are depicted in Figure 1.



(**a**) Scene 3 (**b**) Scene 4

(**c**) Scene 5 (**d**) Scene 6

(**e**) Scene 7 (**f**) Scene 8

Figure 3. Synthetic scenes 3–8. See Table 1 for descriptions.

2.3. Hyperparameter Setting

In the course of this paper, we developed a simple construction worker tracking system using deep learning. We did not aim at building the system to serve as an optimal ready-to-use safety device but to investigate the possibilities of synthetic data for training and evaluation of such approaches in construction site scenarios.

Our system detects pedestrian construction workers in the camera images and tracks them over time in consecutive image frames. For detection, we apply YOLOv3 [28] as it combines classification and detection of the workers in a single CNN. The network was trained on parts of the labeled datasets described in Sections 2.1 and 2.2. Training the network from scratch is unfeasible due to the limited amount of real world data even though we have already made extensive use of YOLO's built-in data augmentation features. Hence, we rely on transfer learning to avoid overfitting the network to our small dataset during training. For this, we trained our network with a mini batch size of 64 based on the Darknet53 model, which has been pre-trained on ImageNet [29]. To quickly adjust the network towards the newly introduced class of pedestrian workers, we began the training with a high learning rate of 0.001. The learning rate was then scaled down by a factor of 0.1 each 3800 epochs until training ended after 10,400 epochs. This facilitates a finer adjustment of the weights so that these converge towards an optimal result. For regularization, we adapted the weights by a momentum of 0.9 and a weight decay of 0.0005.

Besides the hyperparameters directly affecting the learning behavior, YOLO's detection quality and especially its speed highly depend on the given input image size. Scaling down high resolution images may vanish smaller image features relevant for proper detection. Conversely, the larger the input image is, the longer it takes for YOLO to process the entire image. A suitable scaling of high resolution images with regard to the case of application has, thus, to be determined beforehand. By comparing the detector's performance on different input image sizes, we identify the size which yields optimal detection results in accordance with the preservation of real-time applicability. For this, we train and evaluate the detector's performance on various scales of the real world datasets described in Section 2.1. Since the input image size should be a multiple of 32 px because of YOLO's downsampling architecture, we start our tests with a size of 416 × 416 px, which is equivalent to a factor of 13. For further tests, we gradually increase that factor each time by 3 up to an image size of 800 × 800 px. The performances of these test runs are compared by the mAP score as proposed for the evaluation of the COCO Detection Challenge [30]. As depicted in Equation (1), besides averaging over recall values at $i = [0, 0.1, 0.2, \ldots, 1.0]$ with n_i values, this score also averages over different IoU thresholds $j = [0.50, 0.55, \ldots, 0.95]$ in n_j steps. As a consequence, more accurate detection results are rewarded with a higher score.

$$mAP = \frac{1}{n_i n_j} \sum_i \sum_j \text{Precision}(\text{Recall}_{i,j}) \tag{1}$$

For reasonable monitoring or assistance systems, an accurate localization of pedestrian workers on construction sites is desirable. Thus, we choose the COCO mAP score as this score implicitly indicates the localization accuracy.

Although we found that detection accuracy generally rises with increasing image size, the detection speed rigorously drops. At the base image size, we measure a detection time for a single image of 46 ms with a mAP of 76.9 % on the evaluation dataset. At our maximum image size, the detection time increased to 97 ms. This corresponds to about 10 frames per second (fps), which would inhibit a reliable tracking as addressed in Section 2.4. Pointing out a mAP of 86.5 % at still about 15 fps, we decide for an image size of 608 × 608 px as a reasonable tradeoff between detection time and mAP. Using this setup, we are able to analyze at least every second frame at frame rates of up to 30 fps. On the one hand, this speed is still sufficient for the tracking while, on the other hand, we can take advantage of a reasonable detection quality.

2.4. Tracking

After the workers are detected in a video frame, a track is assigned to each of them. For this, we apply Kalman filtering, which relies on a motion model only. An appearance model as it is provided by other tracking approaches would be redundant since our YOLO detector already takes care of this. For the purpose of tracking pedestrian workers, a simple motion model that only includes a worker's two-dimensional location (x, y) and velocity (v_x, v_y) is sufficient. According to this, our motion model describes the transition from one state t to the next state $t + 1$, as shown in Equation (2).

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ v_{x,t+1} \\ v_{y,t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ v_{x,t} \\ v_{y,t} \end{pmatrix} \tag{2}$$

Beginning with an initial detection of a worker, the motion model is used to predict this worker's location in the ensuing video frame. Afterwards, the subsequent frame is evaluated. The workers' locations predicted by the Kalman filter in the preceding frame and the detections made by YOLO in the current frame are, then, matched using the Hungarian method. We use the locations of the matching detections to update the estimation of each track towards the actual worker's position. This prevents the tracks from drifting off. Detections that do not match any pre-existing track will create a new track. This way, our detector serves for both, initializing tracks and providing evidence for existing tracks. Those tracks for which no evidence is found will persist for a maximum of 12 frames without further evidence. From this, we can derive the workers' walking path trajectories as well as their prospective walking directions.

Although the Kalman filter was originally developed for linear tracking, it copes with non-linearity to some extent [31]. However, the pedestrian workers' movements can be highly non-linear. In the short time period between only a few consecutive video frames, though, the workers' movements can be assumed to be almost linear. For this reason, the tracks should be updated at least on every second frame. This drastically decreases the probability of occurrence of non-linearities. Considering this, the Kalman filter has to deal with non-linear state transitions only if workers are not detected for several video frames. An adequate detection beforehand is, thus, a necessary requirement for reliable tracking.

3. Results

We conduct different experiments in order to examine the benefits of synthetically generated data. Our experiments investigate both the training and the evaluation of computer vision-based construction site monitoring systems. For this, we developed a tracking system based on YOLOv3 and Kalman filtering. The data generation and the tracking approach are both described in Section 2.

Our first experiment focuses on the effects of synthetic data on the training of the detector. For this, we contrast the performance of YOLOv3 trained on our real world dataset (see Section 2.1) to a detector additionally trained on different subsets of our synthetic dataset, as illustrated in Section 2.2. The suitability of synthetically generated data for the purpose of evaluating the tracking system's performance is, then, determined in a second experiment, shown in Section 3.2. We examine the similarity between the tracking performances of our system when applied to real world video sequences and to their synthetically generated counterparts.

3.1. Effects of Synthetic Data on the Training

In Section 2.3, we determined the baseline mAP of 86.5 % on real world data at our desired input image size. This allows us to examine the effect of adding supplementary synthetic data during training on the detector's performance. For this, we successively add more and more subsets of our synthetic dataset to the set of training samples. These subsets exhibit different characteristics as described in the following. In the first trial, we add the synthetic scenes 1, 2, and 3 to the existing training set. In total, these scenes show 2130 pedestrian worker instances in 750 images. The amount of provided information is not significantly enhanced by these scenes. It only increases the sheer number of samples as the scenes imitate the real world samples, which have already made up the training dataset beforehand. This results in an increase of the mAP by 1.7 percentage points to a value of 88.2 %. Next, we further extend the training dataset by adding the synthetic scenes 4, 5, and 6, which consist of 1510 images showing 5170 worker instances in total. The compositions of these scenes differ from those of the scenes before, but the environmental conditions remain similar. Trained on the resulting dataset, our detector yields a mAP of 88.7 %. This is an increase of only 0.5 percentage points in comparison to the previously trained detector. By subsequently adding scene 7 containing 2490 worker instances in 620 images, we incorporate different lighting conditions into our training dataset. The scene is brightly illuminated, which leads to large cast shadows. Workers are consistently located in over- and underexposed areas. By this, the mAP drastically increases by 5.2 percentage points to 93.9 %. Further training samples of scene 8, which exhibit environmental conditions similar to scene 7, have only little effect on the detection quality. Although this scene provides another 2490 worker instances in 620 images, the mAP only increases to 94.0 %. The bar graph in Figure 4 summarizes the increase of the mAP over the successively added subsets of our synthetically generated dataset.

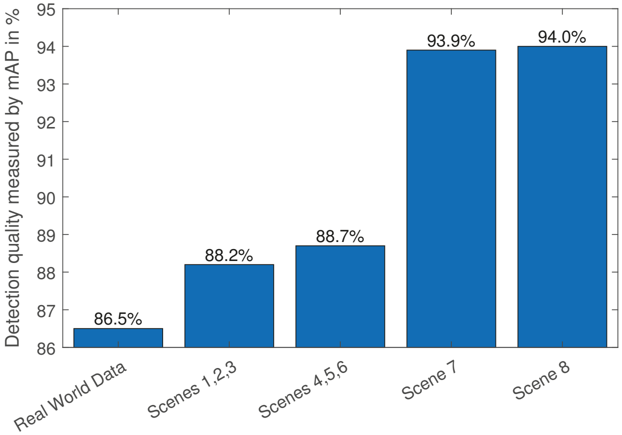


Figure 4. Increase of YOLO's detection quality by successively adding synthetic data samples.

3.2. Tracking System Evaluation

The final detector generated in the previous experiment (see Section 3.1) is embedded into a simple tracking system. Its detections are passed to a Kalman filter to estimate the detected workers' movements, as explicitly described in Section 2.4. By means of the developed tracking system, we investigate the suitability of computer generated 3D scenes for the evaluation of computer vision-based tracking systems. For this, we compare its tracking results on real and synthetic construction site scenarios. Our tracker is compared by means of the two different real world scenarios (see Section 2.1) and their computer generated counterparts described in Section 2.2. From each scenario we extract a continuous 10 second video sequence in which four pedestrian workers move across the scene varying their pace and direction in each sequence. All video sequences also include sudden stops as well as pedestrian workers coming close to and crossing each other. In the real world sequences, ground truth is labeled manually whereas this is done automatically (see Section 2.2) for the computer generated scenes.

We contrast the accuracy of our tracker on each of these video sequences by the average overlap score (AOS) and the center location error ratio (CER) metrics while its robustness is measured by the track length (TL). These metrics are adapted from [32] as shown in Equations (3) to (5).

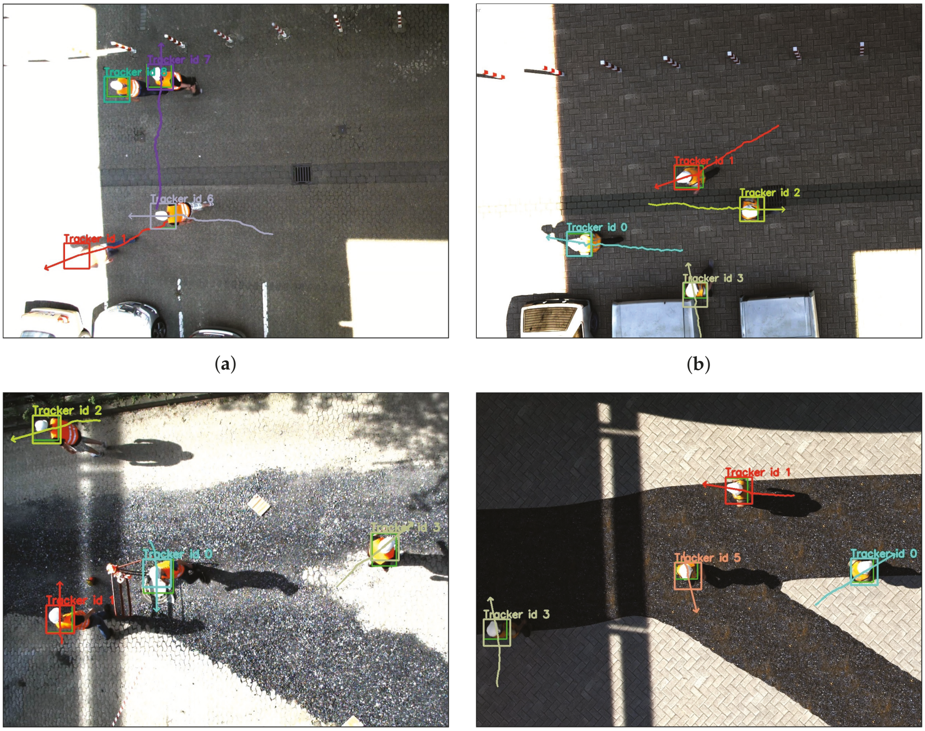
$$\mathrm{AOS} = \frac{1}{n}\sum_{t=1}^{n}\frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}, \tag{3}$$

$$\mathrm{CER} = \frac{1}{n}\sum_{t=1}^{n}\frac{||C_t^G - C_t^T||_2}{\mathrm{size}(A_t^G)}, \tag{4}$$

$$\mathrm{TL} = \frac{n}{N}, \tag{5}$$

where n and N denote the number of video frames in which a worker is tracked and the number of frames in which the worker is present, respectively. The workers' bounding box areas are indicated by A and their centers by C. The superscripts G and T indicate ground truth and tracked bounding boxes, respectively. Lastly, $|| \circ ||_2$ denotes the two dimensional euclidean distance and $\mathrm{size}(\circ)$ represents the two dimensional size of an area.

Tables 2 and 3 summarize the tracking results on the first and second pair of scenes, respectively. As can be seen, in each of the sequences our tracker identifies four tracks. Each of these corresponds to one of the pedestrian workers as illustrated in Figure 5 for all sequences. No tracks are set up mistakenly tracking other objects in the scenes. With an average of 96 % in AOS and a very low CER on both real world scenes, our tracking system yields an accurate performance. In addition, the averages in TL of 97 % and 98 % highlight the systems' robustness. Deviations from the optimal TL are mainly due to the fact that it takes a few frames until sufficient consecutive detections are identified to start a track. In isolated cases, starting a track was delayed since the initial detection was complicated by odd lighting conditions. The measurements' averages of the synthetic scenes are slightly lower than those of the real world scenes, but still demonstrate an accurate performance of the tracking system.



(**a**) (**b**)

(**c**) (**d**)

Figure 5. Tracking results on the real world sequences in (**a**,**c**) and their synthetic counterparts in (**b**,**d**). Ground truth labels are marked as green rectangles whereas differently colored rectangles illustrate tracked workers' positions. Each worker's trajectory is represented by a colored line and the prospective movement is depicted as a colored arrow.

Table 2. Comparison of our system's tracking performance on the first scene from our real dataset and its synthetic counterpart. IDs are assigned randomly and correspond to a single tracked worker each.

	Real Scene 1					**Synthetic Scene 1**				
ID	1	6	7	8	AVG	0	1	2	3	AVG
AOS	0.91	0.96	0.99	0.98	0.96	0.94	0.90	0.94	0.90	0.92
CER	0.0017	0.0015	0.0014	0.0017	0.0016	0.0013	0.0020	0.0018	0.0033	0.0021
TL	0.98	0.92	0.99	0.99	0.97	0.99	0.90	0.94	0.98	0.95

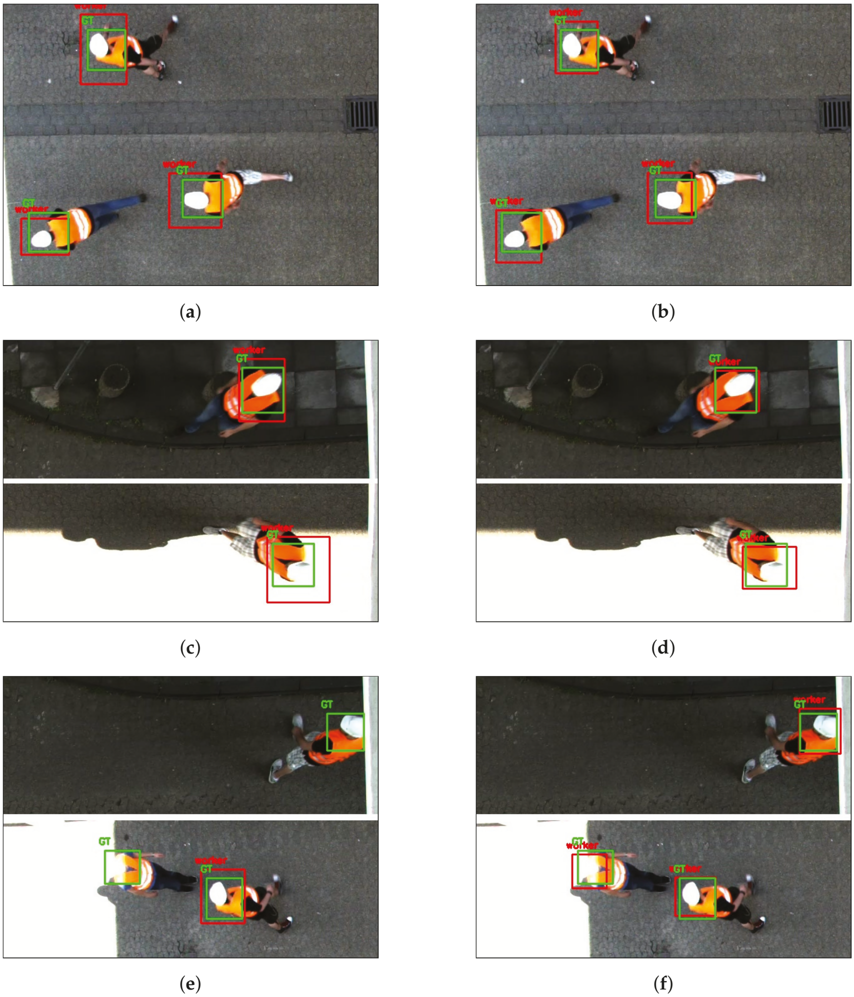
Table 3. Comparison of our system's tracking performance on the second scene from our real dataset and its synthetic counterpart. IDs are assigned randomly and correspond to a single tracked worker each.

	Real Scene 2					**Synthetic Scene 2**				
ID	0	1	2	3	AVG	0	1	3	5	AVG
AOS	0.97	0.97	0.98	0.90	0.96	0.97	0.94	0.91	0.95	0.94
CER	0.0012	0.0027	0.0022	0.0035	0.0024	0.0025	0.0031	0.0038	0.0021	0.0029
TL	0.97	0.98	0.99	0.98	0.98	0.99	0.99	0.91	0.99	0.97

4. Discussion

Our first experiment, shown in Section 3.1, highlights the benefits of using computer generated data for training a detector on construction related scenarios when only little real world data is available. Making use of such data, we boosted our detector's precision from 86.5 % in mAP using only a small real world dataset to 94.0 % by augmenting the dataset with synthetic data. This amounts to an increase of 7.5 percentage points. Nevertheless, the experiment also shows that synthetic data samples have to be chosen carefully in order to properly increase the detection quality. Adding more data samples of similar kind to a small set of training data generally improves the detection quality. This becomes apparent when extending our dataset by the synthetic scenes 1, 2, and 3, which increases the mAP by 1.7 percentage points (see Figure 4). The detection results in Figure 6a,b illustrate that the detected regions become more accurate, which results in a higher mAP score. However, the need for more samples of one specific kind is satiated at a certain amount. The model learned by the detector, then, already considers all relevant features provided by such samples. No further insights can be drawn from such samples. As a consequence, the detection quality languishes as it occurred when adding the scenes 4, 5, and 6 to the training dataset. A further addition of more samples could cause the model to overfit to these training samples, resulting in an even worse performance on the evaluation set. The dataset extension by scene 7 points out that new samples possessing various conditions can further improve the detection results even though these are of synthetic nature. As shown by the comparison of Figure 6c,d, this enables the detector to even precisely identify workers that were only coarsely detected before due to unusual lighting conditions. Furthermore, the large amount of different training samples aggregated from the synthetic scenes 1–7 enhances the detector's generalization ability so that it is able to cope with slight occlusions and partial color changes due to overexposed lighting conditions (see Figure 6e,f). When adding too many samples of a kind, once more the results begin to languish illustrated by extending the dataset by scene 8. These findings show that computer generated data is exceedingly advantageous to augment a dataset by environmental and lighting conditions that do not occur often. Recording training data on construction sites over a long period in order to cover all conditions is a tedious task. Instead, recording only a short real world dataset at certain conditions would be sufficient while particular weather and lighting conditions could be simulated using a 3D environment.

Considering the severe downscaling of the input images, the achieved mAP is already a prominent performance. Due to the scaling factor of about 0.317, the ground truth regions shrink from the original 40–50 px to a size of only 13–16 px in the scaled images. Despite the fact that CNNs generally struggle with the detection of very small objects, a precise detection by means of IoU is difficult. Each pixel offset between the detection and the ground truth label in the scaled image implies an offset of about 3 px in the original image. Accordingly, if the detection is only 1 px off, the best possible IoU decreases to 0.94. An offset of 1 px in both x- and y-direction further decreases the IoU to 0.88 at best. Taking into account that manually labeling ground truth is never accurate to a single pixel, minor deviations between detection and ground truth are inevitable. Thus, an IoU of more than 0.85 can already be assumed to be optimal in this case. With respect to this, we can adapt Equation (1) so that the last bin of j contains the results for all IoU values ranging from 0.85 to 0.95. By this modified measure, our detector achieves a mAP of 99.2 %.



(**a**) (**b**)

(**c**) (**d**)

(**e**) (**f**)

Figure 6. Comparing the detection results of YOLO trained on different datasets. Red rectangles display YOLO's detections whereas green rectangles indicate manually labeled ground truth. (**a**,**c**,**e**) show the results of a detector trained on our real world dataset only. The right column illustrates the detection improvements for the same image sections if particular synthetic data is additionally provided for training. In (**b**), the general accuracy is improved by extending the training dataset by similar data. In (**d**), detections in under- and overexposed areas are improved by adding samples with different lighting conditions. In (**f**), workers can be detected despite slight occlusions and dramatic overexposure after adding a large amount of data samples.

In conclusion, we showed in this experiment that computer generated data is capable of successfully augmenting a construction site related dataset for the purpose of training a CNN. The reasonable choice of synthetic training samples can considerably increase the detection quality. These findings correspond to those in other application areas. It shows that such data is not only advantageous for well structured scenarios like car parks, but also yields reasonable results in crucially more complex environments like construction sites. This is also confirmed by the tracking results on the real world scenes of the second experiment. The simple tracking system based on a detector which was trained on only sparse real world data augmented with synthetic samples already enables suitable monitoring. This further substantiates the use of synthetic data for the training of CNNs in the context of construction sites.

Beyond these findings, the second experiment emphasizes the comparability of real world and synthetic scenes in terms of evaluating vision-based detectors and trackers. The comparison of the tracking results given in Tables 2 and 3 reveals that our tracker acts similar on both kinds of data. On both scenes, the accuracy measured by the AOS and CER on the synthetic sequences is slightly lower than these on real world sequences. This is not necessarily due to the nature of synthetic data but rather associated with the more precise ground truth labels on the synthetic dataset resulting from the automatic labeling method. These labels enclose the workers even closer than those in the manually labeled sequences so that minor deviations during detection result in lower AOS and CER. The comparison of the green ground truth rectangles in Figure 5a,b as well as in Figure 5c,d illustrates this graphically. Nevertheless, on average there is only a deviation of about four percentage points in AOS and a deviation of 0.0005 in CER on the first scene. Similarly, low deviations are given for the second scene with two percentage points in AOS and 0.0005 in CER. These results indicate a reasonable comparability of the tracking performance on real world and synthetic scenarios. Very similar track lengths on both scenes additionally confirm this finding.

The comparison has shown that the evaluation results on real world and computer generated video sequences resemble each other closely. Accordingly, the quality of a tracking system can be deduced on the basis of computer generated video sequences if sufficient real world data cannot be acquired. For construction site scenarios, this is often the case for hazardous situations since intendedly endangering workers should be avoided. Furthermore, weather conditions may appear on which the detector was not explicitly trained. On the basis of the similar evaluation results on synthetic and real world data, we demonstrate the capabilities accompanied by a virtual environment as developed in this paper in an exemplary way. In this example we use a computer generated video to show the evaluation of a tracking system on a risky situation without exposing any worker to a hazard. Furthermore, it illustrates that various environmental conditions can be simulated without the need for tedious repetitive recordings on multiple days. In order to highlight these benefits, we apply our tracking system to a modified version of synthetic scene 1. We change the weather conditions from a sunny to a heavy rainy day and include a crane lifting and pivoting its load above the heads of pedestrian workers. As depicted in Figure 7, again all four workers are tracked by our system despite the rainy weather conditions, which were not explicitly trained. Table 4 shows that only four tracks were assigned, each corresponding to one of the workers. Though, the trackers accuracy slightly decreases. This indicates that the performance of our tracker on a similar real world scenario should basically be still sufficient but its precision might decrease slightly. However, additionally training the detector on data samples of rainy days might counteract this. Furthermore, Table 4 unveils that the tracking is not interrupted even though the workers are temporarily occluded by the crane's load. This demonstrates that the tracking system proposed in this paper is even capable of dealing with certain hazardous situations. By identifying the hazardous area around the crane's load, the crane operator could be warned against approximating pedestrian workers so that risky situations could be prevented. Further tests have to verify whether the system can also cope with larger loads and more difficult weather conditions.

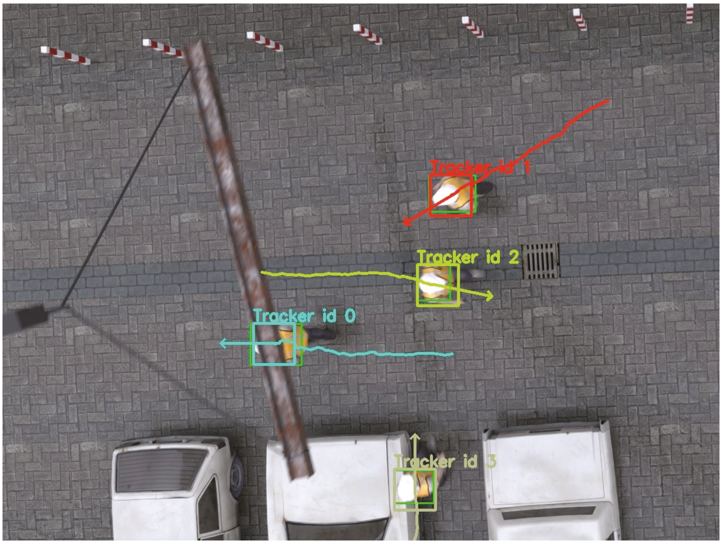


Figure 7. Tracking results on computer generated hazardous situation on a heavy rainy day. Green rectangles denote ground truth labels whereas differently-colored rectangles illustrate the tracked pedestrian workers.

Table 4. Tracking results of our system on synthetic scene 3. IDs are assigned randomly and correspond to a single tracked worker each.

ID	0	1	2	3	AVG
AOS	0.93	0.90	0.94	0.90	0.92
CER	0.0015	0.0033	0.0017	0.0064	0.0032
TL	0.99	0.99	0.99	0.98	0.99

5. Conclusions

Productivity and safety management on construction sites could benefit from monitoring pedestrian workers. Based on recorded walking trajectories of workers provided by a tracking system, the workers' current paths could be assessed and optimized to solve certain tasks more efficiently. These trajectories also reveal the workers' attention with respect to hazards like falling edges or hazardous materials and areas. Owing to this, safety trainings could be tailored explicitly to the needs of the workers. If the localization of workers is conducted live, workers and machine operators could even be warned of looming hazardous situations, which enables them to counteract early. Computer vision methods are more suitable for such a precise tracking of workers all over the site due to various shortcomings arising from the radio-frequency technology used by tag-based alternatives. However, computer vision approaches have to be trained and evaluated on a vast amount of images. Appropriate datasets are rarely publicly available and recording a sufficient amount of data is extremely time-consuming. For this reason, we investigated the use of synthetic data, which is generated from 3D environments.

Besides a small real world dataset, we generated a synthetic dataset that covers diverse environmental and illumination conditions. In order to analyze the usability of data generated from 3D scenarios, we built a simple tracking system. This consists of a YOLOv3 detector identifying pedestrian workers and a Kalman filter tracking those workers in video sequences. In our experiments, we examined the suitability of synthetic data individually for training and evaluation purposes of a computer vision tracking system. First, we iteratively added more and more synthetic data samples to the training dataset of our YOLO detector. Second, we compared the performance of our tracking system on real world and corresponding 3D generated video sequences.

We found that training on synthetic data samples significantly enhances the detection quality. In our experiments, we were able to boost our detector by 7.5 percentage points over a detector trained on a small real world dataset only. Though, the quality of the resulting detector is highly dependent on the choice of decent training samples. As for real world datasets, synthetic samples should also cover various environmental and lighting conditions. Furthermore, we found that a computer vision-based tracker performs very similarly on real world and 3D generated video sequences. Accordingly, the evaluation on synthetically generated scenes can already provide reliable insights regarding the strengths and weaknesses of a tracking system since its performance can be estimated considerably precisely. As a result, a vision-based tracking system can be tested on a variety of synthetically generated situations before being employed on a real construction site. By this, a flawless tracking can be guaranteed even for rare or exceptional situations.

The findings of our experiments are in accordance with those from other application areas, but additionally highlight that synthetic data are capable of modeling even the complex and dynamic environments of construction sites realistically. For construction site-related applications, this validation is relevant to a special degree. Since datasets are typically rare in this field, data augmentation using synthetic data could advance the use of vision-based approaches in the future. In particular, this facilitates incorporating conditionally occurring lighting and weather conditions into a dataset. By simulating, for example, snowfall or bright sunlight, in a virtual environment, data acquisition can be drastically accelerated and existing datasets can easily be completed. The use of synthetic data allows to model any weather and lighting conditions and to change the construction site setup at will. Otherwise, recordings all over the year and on different sites would be necessary to acquire a reasonable dataset. Further time savings and human resources result from the possibility of automatic labeling. Since the positions of all objects in a virtual scene are known exactly, labeling can be done fully automatically and with substantially higher precision than a manual labeling. Via the extension of various environmental conditions and a precise labeling, datasets can be prepared such that an optimal training is ensured. The resulting system can deal with manifold upcoming conditions without having seen these in the real world before. Besides the training of detectors, computer generated data is also valuable for the evaluation of vision systems. Again, virtual environments can be used to simulate a variety of scenarios to test the system on. This is particularly advantageous for events that rarely occur or are hazardous to replicate in the real world. Risky situations can be evaluated this way without exposing anyone to a serious hazard.

In summary, our major conclusions are as follows: We have shown the applicability of synthetically generated data for vision systems in the area of construction sites. Furthermore, we highlighted the benefits of such data for training and evaluation purposes for the underlying machine learning algorithms. As this paper outlines an overview of the possibilities accompanied with synthetic data, future work should investigate either the training or the evaluation phase in more detail. For now, the limits of using synthetic data for both, training and evaluation remain unclear. It should also be determined to what extent the use of synthetic data is beneficial and if certain scenarios cannot be modeled sufficiently. The impact of such data on different machine learning algorithms could be another topic to focus on. Finally, a tracking system optimally trained and tested on a combination of real world and synthetic data should be employed on a real construction site for a large-scale case study.

Author Contributions: Conceptualization, M.N. and P.H.; data curation, P.H.; investigation, M.N. and P.H.; methodology, M.N.; project administration, M.N.; resources, M.N. and P.H.; software, M.N.; supervision, M.K.; validation, M.N.; visualization, M.N. and P.H.; writing—original draft, M.N. and P.H.; writing—review and editing, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The motion capture data used in this project was obtained from mocap.cs.cmu.edu. The CMU database was created with funding from NSF EIA-0196217. Furthermore, we acknowledge the support by the DFG Open Access Publication Funds of the Ruhr-Universität Bochum.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cheng, T.; Yang, J.; Teizer, J.; Vela, P.A. Automated Construction Resource Location Tracking to Support the Analysis of Lean Principles. In Proceedings of the Annual Conference of the International Group for Lean Construction, Haifa, Israel, 14–16 July 2010; pp. 643–653.
2. Teizer, J.; Allread, B.S.; Fullerton, C.E.; Hinze, J. Autonomous Pro-Active Real-Time Construction Worker and Equipment Operator Proximity Safety Alert System. *Autom. Constr.* **2010**, *19*, 630–640. [CrossRef]
3. Nasr, E.; Shehab, T.; Vlad, A. Tracking Systems in Construction: Applications and Comparisons. In Proceedings of the Annual Conference of Associated Schools of Construction, San Luis Obispo, CA, USA, 10–13 April 2013; pp. 9–13.
4. Riaz, Z.; Edwards, D.J.; Thorpe, A. SightSafety: A Hybrid Information and Communication Technology System for Reducing Vehicle/Pedestrian Collisions. *Autom. Constr.* **2006**, *15*, 719–728. [CrossRef]
5. Khoury, H.M.; Kamat, V.R. High-Precision Identification of Contextual Information in Location-Aware Engineering Applications. *Adv. Eng. Inform.* **2009**, *23*, 483–496. [CrossRef]
6. Xie, P.; Petovello, M.G. Measuring GNSS Multipath Distributions in Urban Canyon Environments. *IEEE Trans. Instrum. Meas.* **2015**, *64*, 366–377. [CrossRef]
7. Chae, S.; Yoshida, T. Application of RFID Technology to Prevention of Collision Accident with Heavy Equipment. *Autom. Constr.* **2010**, *19*, 368–374. [CrossRef]
8. Carbonari, A.; Giretti, A.; Naticchia, B. A Proactive System for Real-Time Safety Management in Construction Sites. *Autom. Constr.* **2011**, *20*, 686–698. [CrossRef]
9. Pradhan, A.; Ergen, E.; Akinci, B. Technological Assessment of Radio Frequency Identification Technology for Indoor Localization. *J. Comput. Civ. Eng.* **2009**, *23*, 230–238. [CrossRef]
10. Juels, A. RFID Security and Privacy: A Research Survey. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 381–394. [CrossRef]
11. Han, S.; Lee, S.; Peña-Mora, F. Vision-Based Detection of Unsafe Actions of a Construction Worker: Case Study of Ladder Climbing. *J. Comput. Civ. Eng.* **2013**, *27*, 635–644. [CrossRef]
12. Yang, J.; Shi, Z.; Wu, Z. Vision-Based Action Recognition of Construction Workers Using Dense Trajectories. *Adv. Eng. Inform.* **2016**, *30*, 327–336. [CrossRef]
13. Park, M.W.; Brilakis, I. Construction Worker Detection in Video Frames for Initializing Vision Trackers. *Autom. Constr.* **2012**, *28*, 15–25. [CrossRef]
14. Park, M.W.; Brilakis, I. Continuous Localization of Construction Workers via Integration of Detection and Tracking. *Autom. Constr.* **2016**, *72*, 129–142. [CrossRef]
15. Luo, X.; Li, H.; Wang, H.; Wu, Z.; Dai, F.; Cao, D. Vision-Based Detection and Visualization of Dynamic Workspaces. *Autom. Constr.* **2019**, *104*, 1–13. [CrossRef]
16. Vierling, A.; Sutjaritvorakul, T.; Berns, K. Crane Safety System with Monocular and Controlled Zoom Cameras. In Proceedings of the International Symposium on Automation and Robotics in Construction, Berlin, Germany, 20–25 July 2018. [CrossRef]
17. Parker, S.P. *McGraw-Hill Dictionary of Scientific and Technical Terms*; McGraw-Hill Education: New York, NY, USA, 2003.
18. Jaderberg, M.; Simonyan, K.; Vedaldi, A.; Zisserman, A. Reading Text in the Wild with Convolutional Neural Networks. *Int. J. Comput. Vis.* **2016**, *116*, 1–20. [CrossRef]
19. Gupta, A.; Vedaldi, A.; Zisserman, A. Synthetic Data for Text Localisation in Natural Images. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2315–2324.
20. Tremblay, J.; Prakash, A.; Acuna, D.; Brophy, M.; Jampani, V.; Anil, C.; To, T.; Cameracci, E.; Boochoon, S.; Birchfield, S. Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 969–977.

21. Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3234–3243.
22. Atapour-Abarghouei, A.; Breckon, T.P. Real-Time Monocular Depth Estimation Using Synthetic Data With Domain Adaptation via Image Style Transfer. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2800–2810. [CrossRef]
23. Richter, S.R.; Vineet, V.; Roth, S.; Koltun, V. Playing for Data: Ground Truth from Computer Games. In *Computer Vision—ECCV 2016*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 102–118._7. [CrossRef]
24. Tschentscher, M.; Prus, B.; Horn, D. A Simulated Car-Park Environment for the Evaluation of Video-Based on-Site Parking Guidance Systems. In Proceedings of the IEEE Intelligent Vehicles Symposium, Redondo Beach, CA, USA, 11–14 June 2017; pp. 1571–1576. [CrossRef]
25. Horn, D.; Houben, S. Evaluation of Synthetic Video Data in Machine Learning Approaches for Parking Space Classification. In Proceedings of the IEEE Intelligent Vehicles Symposium, Changshu, China, 26–30 June 2018; pp. 2157–2162. [CrossRef]
26. Zhu, J.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2223–2232. [CrossRef]
27. Wang, Q.; Gao, J.; Lin, W.; Yuan, Y. Learning From Synthetic Data for Crowd Counting in the Wild. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–21 June 2019; pp. 8190–8199. [CrossRef]
28. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
29. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami, Florida, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
30. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. COCO—Common Objects in Context. Available online: http://cocodataset.org/#detection-eval (accessed on 10 March 2020).
31. Kim, H.; Kim, K.; Kim, H. Vision-Based Object-Centric Safety Assessment Using Fuzzy Inference: Monitoring Struck-By Accidents with Moving Objects. *J. Comput. Civ. Eng.* **2016**, *30*, 04015075. [CrossRef]
32. Xiao, B.; Zhu, Z. Two-Dimensional Visual Tracking in Construction Scenarios: A Comparative Study. *J. Comput. Civ. Eng.* **2018**, *32*, 04018006. [CrossRef]

applied sciences

MDPI

Article

Environment Classification for Unmanned Aerial Vehicle Using Convolutional Neural Networks

Carlos Villaseñor [1], Alberto A. Gallegos [2], Javier Gomez-Avila [1], Gehová López-González [2], Jorge D. Rios [1] and Nancy Arana-Daniel [1,*]

[1] Department of Computer Science, University of Guadalajara, 1421 Marcelino García Barragán, Guadalajara 44430, Jalisco, Mexico; carlos.villasenor@academicos.udg.mx (C.V.); jenrique.gomez@academicos.udg.mx (J.G.-A.); jorge.rarranaga@academicos.udg.mx (J.D.R.)

[2] Department of Artificial Intelligence, Hydra Technologies of Mexico, 6503 Vallarta Eje Poniente, Guadalajara 45010, Jalisco, Mexico; agallegos@hydra-technologies.com (A.A.G.); glopez@hydra-technologies.com (G.L.-G.)

* Correspondence: nancy.arana@academicos.udg.mx

Received: 18 June 2020; Accepted: 16 July 2020; Published: 20 July 2020

Featured Application: The approach presented in this paper is implemented in an autonomous UAV to provide the ability to change its path according to ground position and weather conditions, since sustaining an aircraft when flying through a dense cloud is not possible.

Abstract: Environment classification is one of the most critical tasks for Unmanned Aerial Vehicles (UAV). Since water accumulation may destabilize UAV, clouds must be detected and avoided. In a previous work presented by the authors, Superpixel Segmentation (SPS) descriptors with low computational cost are used to classify ground, sky, and clouds. In this paper, an enhanced approach to classify the environment in those three classes is presented. The proposed scheme consists of a Convolutional Neural Network (CNN) trained with a dataset generated by both, an human expert and a Support Vector Machine (SVM) to capture context and precise localization. The advantage of using this approach is that the CNN classifies each pixel, instead of a cluster like in SPS, which improves the resolution of the classification, also, is less tedious for the human expert to generate a few training samples instead of the normal amount that it is required. This proposal is implemented for images obtained from video and photographic cameras mounted on a UAV facing in the same direction of the vehicle flight. Experimental results and comparison with other approaches are shown to demonstrate the effectiveness of the algorithm.

Keywords: cloud detection; superpixel segmentation; convolutional neural networks; support vector machines

1. Introduction

Unmanned Aerial Vehicles (UAVs) have gained popularity in the last decades due to their capability for moving in three-dimensional space. UAVs were first used for military purposes. However, they are now used for surveillance, research, monitoring, and search and rescue activities [1]. These kinds of vehicles are suited for situations that are too dangerous and hazardous where direct monitoring is not humanly possible [2].

One of the challenges of UAV is the loss of communication with the remote pilot. For this reason, it is necessary to provide the vehicle with a certain level of autonomy to maintain flight in such scenarios. A UAV must be able to adapt and change its path according to ground position and weather conditions, since sustaining an aircraft when flying through a dense cloud is not possible [3].

Given weather indicators that allow the detection of clouds, can be seen from long distances [4]; it is possible to develop an intelligent system capable of avoiding them.

Cloud detection is a very challenging task; each big water cluster has a unique amorphous shape, which is continuously changing; making it impossible to extract characteristic features to be tracked with some descriptor such as with Speeded Up Robust Features (SURF) [5], then, other methods to extract information are needed, such as segmentation based on color, texture, and illumination [6–9].

In Reference [10], several simple-to-implement descriptors with linear computational costs are presented, showing a good training and generalization. Results from a video camera mounted on a UAV reported satisfactory results for two and three class classification in real-time.

Our proposed scheme describes and implements an approach to classify three elements of the environment (ground, sky, and clouds), using Superpixel Segmentation (SPS) and Support Vector Machine (SVM) to pre-train a Convolutional Neural Network (CNN), which is a form of deep learning model, trained end-to-end from raw pixel intensity values to classifier outputs. The spatial structure of the images makes it suitable to work with this kind of networks, setting connectivity between the filters (or layers) and the parameter sharing, and discrete convolutions [11].

The used images in this work were captured by a camera mounted on a UAV provided by Hydra Technologies of Mexico®; an example of an obtained image is shown in Figure 1.

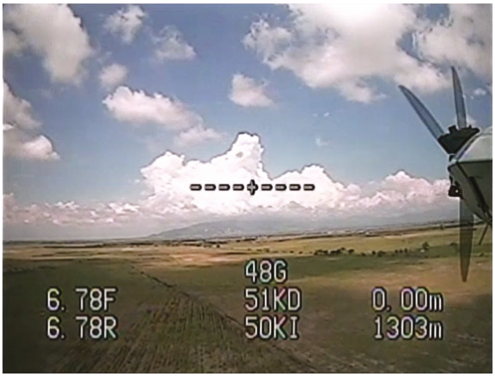


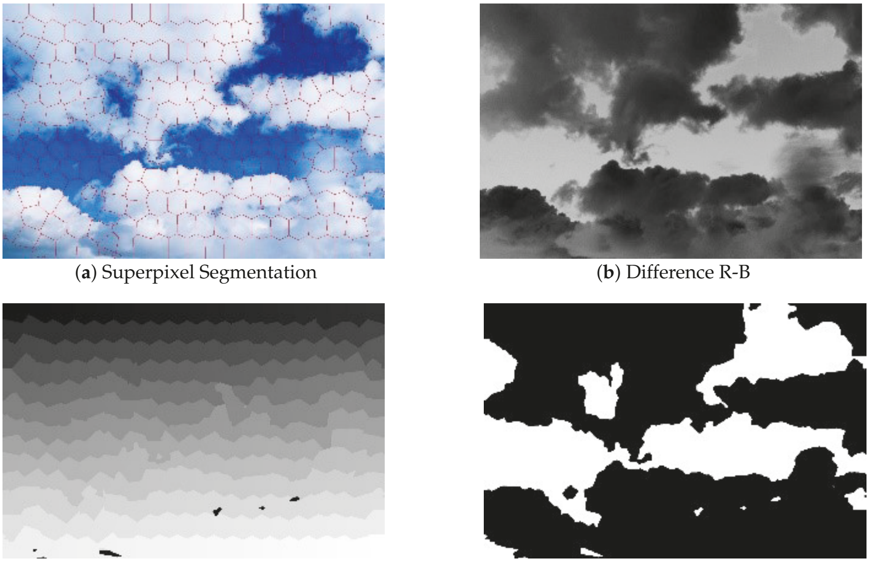
Figure 1. Captured image of a video stream from a camera mounted on an unmanned aerial vehicle (UAV). The image resolution is 720 width and 480 height at a 30 frames per second.

The outline of this papers is as follows: In Section 2, related work is presented. Section 3 presents a brief description of the used SVM whose output is used to pre-train the CNN. Section 4 presents the descriptors based on SPS methodology. In Section 5, the CNN architecture is described. Experimental results are presented in Section 6 and important conclusions are discussed in Section 7.

2. Related Work

Most of the research done on cloud detection is ground-based, where clouds are captured with instruments that obtain continuous all-sky images at pre-defined time intervals [12,13]. For a UAV, it is impossible to keep these conditions since the update intervals of information need to be shorter. Moreover, algorithms should not have a high computational cost, because onboard computers may not have the same processing power and memory capabilities as an off-board station. Also, a computer with high processing power in a UAV would require a higher demand for energy, which would require batteries with higher capabilities increasing the UAV weight, affecting the fuel consumption of the aircraft negatively.

Other works solve the problem of object identification using an undirected graph [14]. Computing the graph association matrix could be computationally expensive; in the worst-case scenario, it is a problem of $O(n^2)$ complexity [7,14]. These approaches are not suitable for real-time applications working with high definition images [9]. In Reference [13], an automatic cloud detection for all-sky images using SPS is presented; the result and implementation of this algorithm shown in Figure 2. It can be seen that even if it is a good approximation, some information is lost in the final result. Considering these results and the computational complexity of the algorithm, it may not be suitable for these kinds of applications.


(**a**) Superpixel Segmentation

(**b**) Difference R-B

(**c**) Threshold matrix

(**d**) Detection result

Figure 2. Steps of the algorithm described in Reference [11].

On the other hand, algorithms based on image matting [6,8,15] try to reduce computational complexity. These algorithms extract foreground objects in images, but they are not easy to implement and take long processing time [9]. In these approaches, the algorithm distinguishes only between two classes (sky or ground), and it is difficult to add more classes.

Recently, deep learning techniques have been used to solve many computer vision tasks [14,16–20]. In particular, CNNs are good image classifiers [21–26]. Approaches like the ones presented in References [27,28] use CNNs that are trained to predict a class for each pixel. In contrast, this paper employs a segmentation on top of a CNN to label these clusters of pixels as the clean sky, clouds and ground.

3. Support Vector Machines

Vapnik introduced support vector machines in 1995, and they are widely used in classification tasks because of its simplicity and the convexity of the function to optimize [29]. Classification is treated as an optimization problem; the aim is to minimize a risk function R and maximize the separation between classes as represented by

$$w^* = \arg\min_{w \in \Re^D} F(w) = \frac{1}{2}\|w\|^2 + \zeta R(w), \tag{1}$$

where w is a normal vector orthogonal to the separating hyperplane, $\frac{1}{2}\|w\|^2$ is a quadratic regularization term, and $\zeta > 0$ is a fixed constant that limits the risk function. Equation (1) can be expressed using Lagrange multipliers as follows

$$\begin{aligned} \arg\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \psi_i \psi_j \Omega\left(\beta_i, \beta_j\right) \\ w^* = \sum_{i=1}^{n} \alpha_i \psi_i \beta_i, \end{aligned} \tag{2}$$

subject to $0 \leq \alpha_i \leq \zeta$ and $\sum_{i=1}^{n} \alpha_i \psi_i = 0$. Where $(\beta_i, \psi_i)_i^n$ is a training set, from which β_i is an n-dimensional input vector and ψ_i its corresponding label. Notice that α_i are Lagrange multipliers and $\Omega(\beta_i, \beta_j)$ is the value of the kernel matrix Ω defined by the inner product $\phi\left(\beta_i\right) \cdot \phi\left(\beta_j\right)$, where ϕ is a non-linear mapping to a high dimensional space. The advantage of using this dual formulation is the use of kernels that introduce the feature space by implicitly mapping the input data into a higher-dimensional space where non-linearly separable data can be linearly separable [30,31].

A CNN requires a massive amount of training data; this task is usually tedious for a human. In that sense, the data used to pre-train the network has been created by an human expert and a SVM that classifies an image segmented with superpixels, that is, sub-areas represented by only a descriptor instead of having several values for every pixel in the sub-area.

4. Descriptors

Most of the descriptors are developed to classify only two classes and cannot be naturally scaled to m different classes. The descriptors presented in this section have linear complexity $O(n)$, and a descriptor capable of increasing the number of classes to three is proposed.

4.1. Descriptors Based on Superpixel Segmentation and Histogram

In this section, three descriptors that use their histograms as features are described. Three images must be obtained to construct the required descriptors. Let (R, G, B) be the channels red, green, and blue, respectively; the descriptors will be obtained from $R - B$, R/B, and RGB images. Cloud detection algorithms commonly use color to determine if a region of the image is a cloud. Cloud particles have a similar dispersion of B and R intensity, whereas clear sky presents more B than R intensity [12,13].

For N pixels, M superpixels will be generated based on color similarity and proximity using Simple Linear Iterative Clustering (SILC) [32] in CIELAB color space. SILC initializes M clusters centers $C_m = [l_m, a_m, b_m, x_m, y_m]^T$ on a regular grid space, where (l, a, b) is the color vector in CIELAB space and (x, y) are the pixel coordinates. Each superpixel has an approximate size of N/M and the center will be located every $S = \sqrt{N/M}$.

SILC computes a distance D between pixel i and its nearest cluster center C_m

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{s}\right) r^2}, \tag{3}$$

where $r \in [1, 40]$ is a constant that allows pondering between color similarity and spatial proximity, d_c and d_s are defined by

$$d_c = \sqrt{\left(l_j - l_i\right)^2 + \left(a_j - a_i\right)^2 + \left(b_j - b_i\right)^2} \tag{4}$$

$$d_s = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2}. \tag{5}$$

The clusters are adjusted to take the value of the main vector of the pixels in C_m, and a residual error E between the new cluster center and previous centers is computed using L_2 norm. The algorithm stops when E reaches a certain threshold.

The descriptor β of the superpixel k is obtained from a histogram of 16 values for each superpixel in $R-B$ and R/B images. The intensity value of pixel $i \in k$ is divided by 16 and rounded downward to its nearest integer value. In the case of the RGB image, a histogram for each channel is obtained.

4.2. Superpixel Segmentation with Gabor Filter

For this approach, a pre-processing step is needed and is showed in Figure 3. Since clouds enhance the $R-B$ difference, this image has been used, and its histogram has been normalized. Gaussian blur has been applied to reduce noise, before the binarization with Otsu's method [33], which is obtained by solving

$$\sigma_b^2\,(t) = P_0 P_1 (\mu_0 - \mu_1)^2, \tag{6}$$

where P_0 and P_1 are class probabilities obtained from a histogram L and separated by a threshold t; and μ_0 and μ_1 are the means of the classes. This is represented by Equations (7)–(10):

$$P_0\,(t) = \sum_{i=0}^{t-1} p\,(i) \tag{7}$$

$$P_1\,(t) = \sum_{i=t}^{L-1} p\,(i) \tag{8}$$

$$\mu_0\,(t) = \sum_{i=0}^{t-1} i\,\frac{p(i)}{P_0} \tag{9}$$

$$\mu_1\,(t) = \sum_{i=t}^{L-1} i\,\frac{p(i)}{P_1}. \tag{10}$$

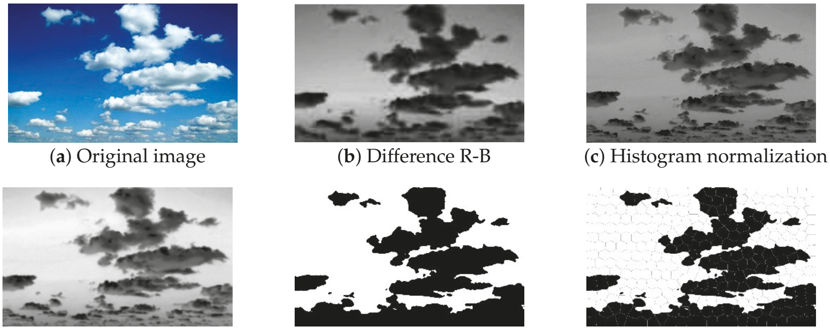
At this step, it is easy to classify clean-sky from clouds; however, as can be seen in Figure 4, it is not possible to make a distinction between clouds and ground. Because of this, it is necessary to use another descriptor capable of distinguishing between them. In this case, the Gabor filter [34] is applied to the original image to get the descriptor because of its ability to permit texture representation and discrimination. The filter has a strong response with structures in the image that have the same direction [35]. The following two-dimensional Gabor functions are used:

$$g_{\lambda,\Theta,\rho}\,(x,y) = e^{-\left(\left(x'^2+\gamma^2 y'^2\right)/2\sigma^2\right)} \cos\left(2\pi\frac{x'}{\lambda}+\rho\right). \tag{11}$$

$$x' = x\cos\Theta + y\sin\Theta \tag{12}$$

$$y' = -x\cos\Theta + y\sin\Theta, \tag{13}$$

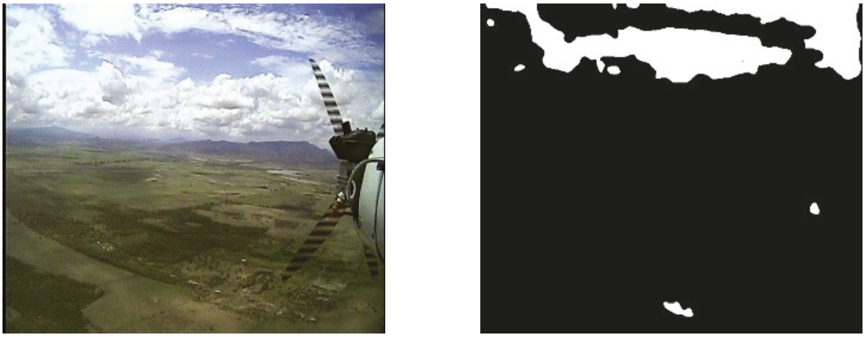
where λ is the wavelength, Θ is the orientation, ρ is the phase offset, γ is the aspect ratio, and $\sigma = 0.56\lambda$ is the standard deviation.



(**a**) Original image (**b**) Difference R-B (**c**) Histogram normalization

(**d**) Gaussian blur (**e**) Otsu's binarization (**f**) Superpixel Segmentation

Figure 3. Steps to generate the proposed descriptor for three classes (ground, cloud and sky).



(**a**) Original image (**b**) Binary image

Figure 4. Clouds and ground classes cannot be easily distinguished.

Four Gabor filters are calculated for $\Theta \in (\pi/4, \pi/(2, 3\pi/(4, \pi)))$. The filtered images are converted to grayscale, and the mean of the values of the image is added to the descriptor. The variance of superpixel k, in each Gabor filtered image is calculated and added to the descriptor β_k. Moreover, spatial information has been included in the descriptor since ground superpixels will have lower spatial values, while clouds superpixels will have higher spatial values.

5. Convolutional Neural Networks

CNNs are commonly used for processing data contained in a matrix or grid, such as images, that are represented by a 2D matrix. Their name comes from the mathematical operation called convolution, which is an operation on two functions to produce a third function that expresses how one of them is modified by the other. In computer vision and image processing, the convolution operation is used to reduce noise and enhance features in images.

Let us suppose that $s(t)$ is the output of the convolution; the operation is given by

$$s\,(t) = \int l\,(a\,)\,h\,(t-a)\,da, \tag{14}$$

where function l is the output of a sensor (and usually referred to as the input in CNN terminology), h is a weighting function (also known as the kernel), a is the age of the measurement. The convolution is commonly denoted with an asterisk as follows

$$s\,(t) = (l * h)\,(t)\,. \tag{15}$$

This data is usually discretized, and if time t can only take integer values then it is possible to define the convolution as a discrete operation as follows

$$s\left(t\right)=\sum_{a=-\infty}^{\infty}l\left(a\;\right)h\left(t-a\right). \tag{16}$$

The input and the output are multidimensional arrays, and every element must be explicitly stored separately. It is assumed that every element out of the set of points, for which the values are stored, is zero; therefore, the infinite summation can be implemented over a finite number of array elements, and also, it can be used over more than one axis at a time. Let I be a two-dimensional image, K a two-dimensional kernel, the convolution for images is given by

$$S\left(i,j\right)=\left(I*K\right)\left(i,j\right)=\sum_{m}\sum_{n}I\left(m,n\right)K\left(i-m,j-n\right) \tag{17}$$

and can graphically be described, as shown in Figure 5.

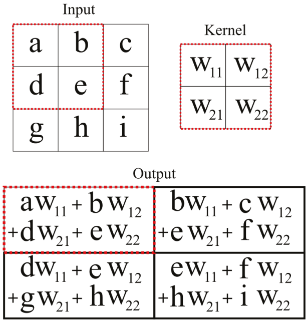

Input
a b c
d e f
g h i
Kernel
w_{11} w_{12}
w_{21} w_{22}
Output
$aw_{11}+bw_{12}+dw_{21}+ew_{22}$
$bw_{11}+cw_{12}+ew_{21}+fw_{22}$
$dw_{11}+ew_{12}+gw_{21}+hw_{22}$
$ew_{11}+fw_{12}+hw_{21}+iw_{22}$

Figure 5. Graphical description of convolution operation.

The convolution presents two properties that can help to improve a machine learning system—sparse interactions and parameter sharing [36].

Due to its sparse interactions, it is necessary to store fewer parameters and fewer operations; however, units in the deeper layers may indirectly interact with a more significant portion of the input and describe more complicated interactions between pixels, as described in Figure 6.

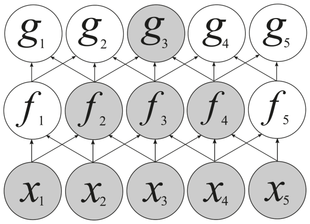


Figure 6. Description of sparse interactions. Even if direct connections seem to be sparse, more units at deeper layers are indirectly connected.

A CNN consist of three steps. First, several convolutions in parallel produce a set of linear activations. Then, a detector step is implemented, where nonlinear activation functions take the linear activations as the argument. Finally, a pooling function is used to modify the output of the layer, making the representation invariant to small translations of the input [36].

Environment Classification with CNN

CNNs have demonstrated effectiveness in image recognition, segmentation, and detection [11]. The architecture of the network is shown in Figure 7. Each layer uses a Rectified Linear Unit (ReLU) function for their activation; except for the last one, whose activation function is a sigmoid, and is given by $f(x) = 1/(1 + e^{-x})$.

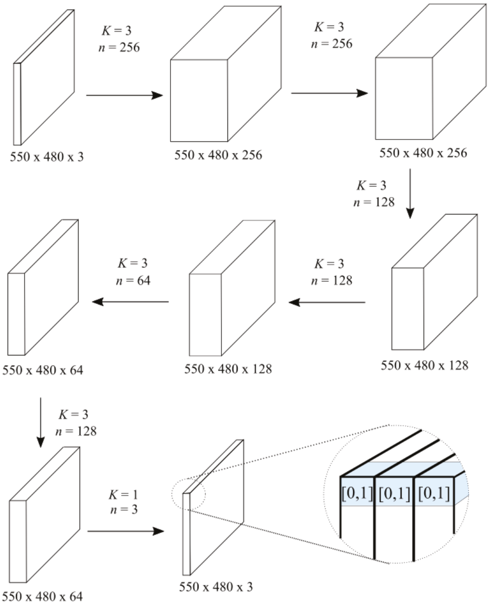


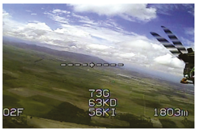
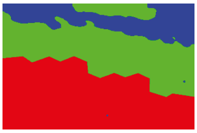
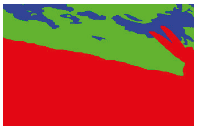
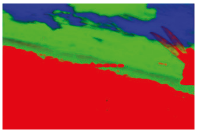
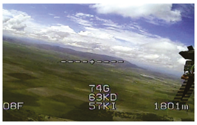
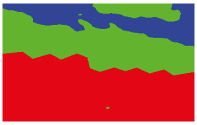
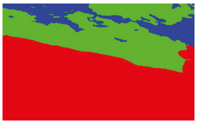
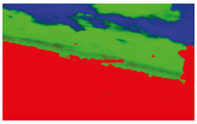
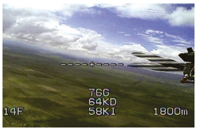
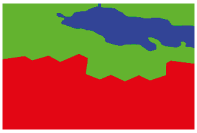
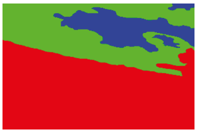
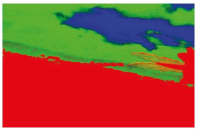
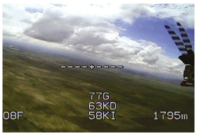
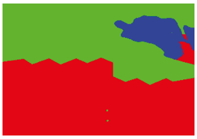
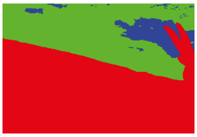
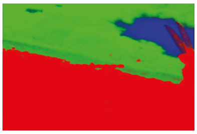
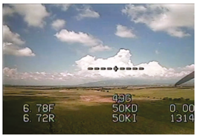
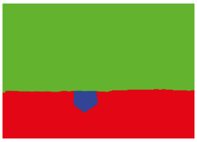
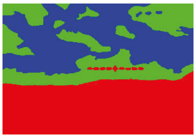
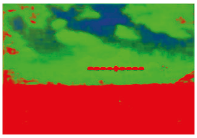
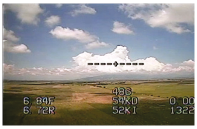
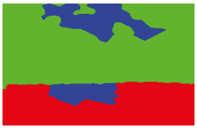
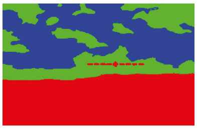
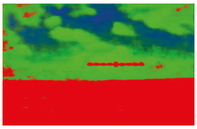
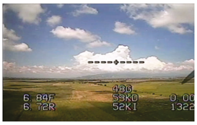
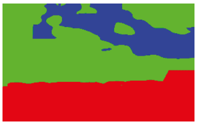
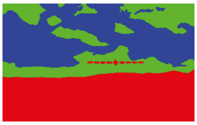
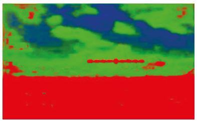
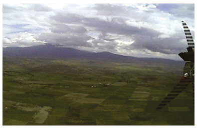
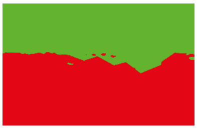
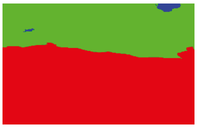
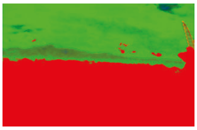
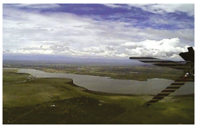
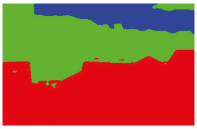
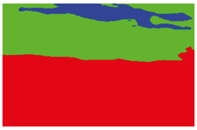
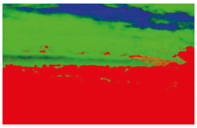
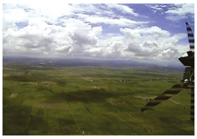
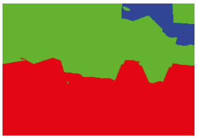
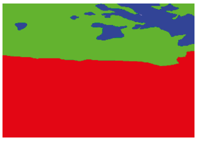
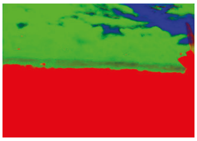
Figure 7. Convolutional Neural Network (CNN) architecture. In all cases stride = 1 and a zero-padding = 1. The output image has three channels (for sky, cloud, and ground), and each pixel is labeled based on its three channels values.

CNN is a class of deep learning model that requires a large quantity of data to be trained. In practice, it is relatively rare to access large data sets, and it is a tedious task for a human to generate them [21]. In this work, one part of this data is generated by the classification of the superpixels made by the SVM; nevertheless, training the CNN only with SVM information would make the CNN learn from a Support Vector Machine. Another set of training data was provided by a human expert to avoid this behavior. Finally, the training data were artificially enlarged using data augmentation.

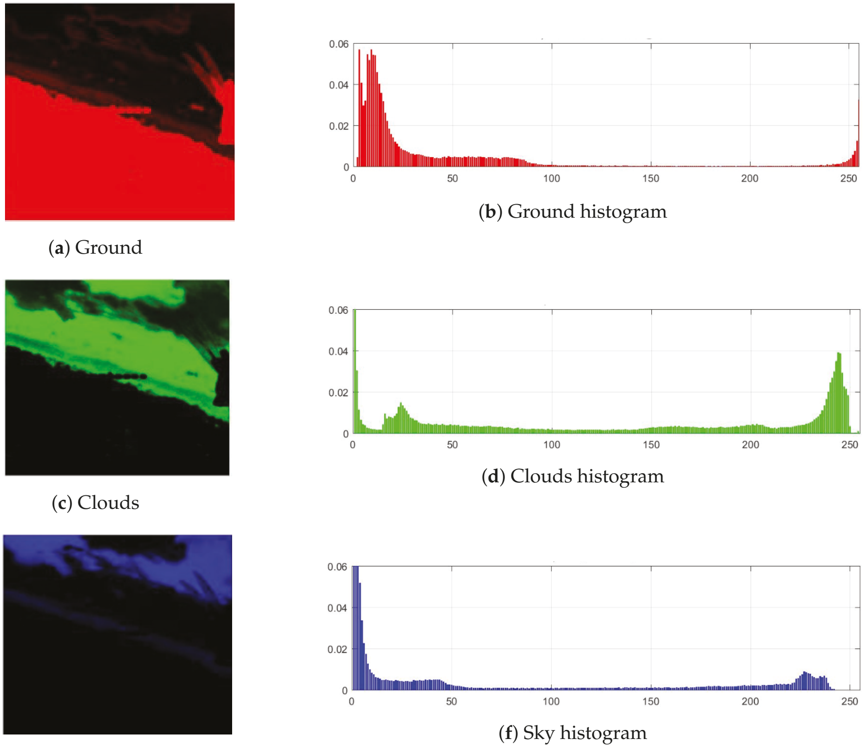
6. Experimental Results

In this section, results of proposal are presented, the pre-train step is carried out with 1000 images provided by an SVM. Then, only twenty ground truth images classified by a human expert are used for supervised training. Table 1 shows ten test images used to demonstrate the effectiveness of the proposed algorithm. These photos were taken from three different flights at a fixed altitude, but different in each flight, and different weather conditions. Although they are not consecutive frames, pictures from rows 5 to 7 were taken from a straight and level flight; and there is little difference between them, however, the SPS-SVM clearly presents a different classification between these images. Additionally, the data set is artificially enlarged, applying geometric transformations to the training set.

Table 1. Test results. The first column shows the original image. The second column shows the Superpixel Segmentation (SPS)–Support Vector Machine (SVM) classification. The ground truth, generated by a human is shown in the third column. In the fourth column, the classification made by the CNN is presented.

Original Image	SPS-SVM Classification	Human Labeled	CNN Output
			
			
			
			
			
			
			
			
			
			

For each pixel, the CNN outputs the probability of belonging to each class. By using these probabilities as pixel intensities, we form grayscale images in Figure 8. Their histogram are also shown. Moreover, the probabilities of each class are scaled and presented in Figure 9 to demonstrate which pixels activate the output layer for each class.



(**a**) Ground

(**b**) Ground histogram

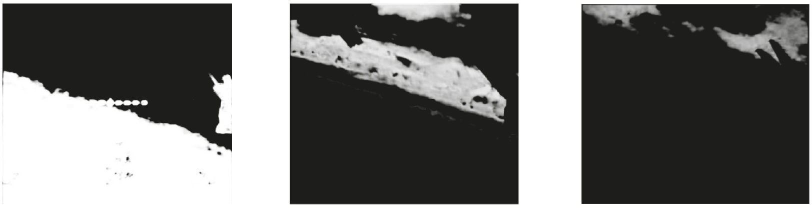
(**c**) Clouds

(**d**) Clouds histogram

(**e**) Sky

(**f**) Sky histogram

Figure 8. Pixel distribution for each class.



(**a**) Scaled ground

(**b**) Scaled clouds

(**c**) Scaled sky

Figure 9. Scaled probabilities for each class.

To display a better visualization of the performance, Table 2 shows the confusion matrices of both approaches. These matrices compare the prediction of the algorithm with the ground truth. The closer it gets to an identity matrix, the less the algorithm gets confused between classes.

Table 2. Confusion Matrices comparison. (S: Sky, C: Cloud, G: Ground).

Test	SPS-SVM vs. Human	CNN vs. Human
1	Predicted True $\begin{array}{cccc} 0.983 & 0.017 & 0.000 & S \\ 0.202 & 0.798 & 0.000 & C \\ 0.015 & 0.223 & 0.763 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.976 & 0.024 & 0.000 & S \\ 0.149 & 0.848 & 0.003 & C \\ 0.013 & 0.116 & 0.871 & G \\ S & C & G & \end{array}$
2	Predicted True $\begin{array}{cccc} 0.979 & 0.021 & 0.000 & S \\ 0.176 & 0.824 & 0.000 & C \\ 0.000 & 0.159 & 0.841 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.972 & 0.002 & 0.000 & S \\ 0.134 & 0.863 & 0.003 & C \\ 0.000 & 0.098 & 0.902 & G \\ S & C & G & \end{array}$
3	Predicted True $\begin{array}{cccc} 0.903 & 0.097 & 0.000 & S \\ 0.058 & 0.917 & 0.025 & C \\ 0.000 & 0.130 & 0.872 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.897 & 0.103 & 0.000 & S \\ 0.039 & 0.912 & 0.048 & C \\ 0.000 & 0.085 & 0.915 & G \\ S & C & G & \end{array}$
4	Predicted True $\begin{array}{cccc} 0.878 & 0.121 & 0.001 & S \\ 0.030 & 0.962 & 0.008 & C \\ 0.019 & 0.121 & 0.860 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.854 & 0.146 & 0.000 & S \\ 0.013 & 0.983 & 0.004 & C \\ 0.016 & 0.090 & 0.894 & G \\ S & C & G & \end{array}$
5	Predicted True $\begin{array}{cccc} 0.000 & 1.000 & 0.000 & S \\ 0.000 & 1.000 & 0.000 & C \\ 0.039 & 0.129 & 0.832 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.192 & 0.806 & 0.002 & S \\ 0.000 & 0.907 & 0.093 & C \\ 0.000 & 0.038 & 0.962 & G \\ S & C & G & \end{array}$
6	Predicted True $\begin{array}{cccc} 0.282 & 0.718 & 0.000 & S \\ 0.002 & 0.998 & 0.000 & C \\ 0.127 & 0.148 & 0.724 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.271 & 0.728 & 0.001 & S \\ 0.000 & 0.915 & 0.085 & C \\ 0.000 & 0.055 & 0.945 & G \\ S & C & G & \end{array}$
7	Predicted True $\begin{array}{cccc} 0.524 & 0.476 & 0.000 & S \\ 0.011 & 0.983 & 0.006 & C \\ 0.000 & 0.129 & 0.871 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.418 & 0.569 & 0.013 & S \\ 0.000 & 0.892 & 0.108 & C \\ 0.000 & 0.048 & 0.952 & G \\ S & C & G & \end{array}$
8	Predicted True $\begin{array}{cccc} 0.000 & 0.999 & 0.001 & S \\ 0.000 & 0.995 & 0.005 & C \\ 0.000 & 0.103 & 0.897 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.087 & 0.912 & 0.001 & S \\ 0.000 & 0.984 & 0.016 & C \\ 0.000 & 0.127 & 0.873 & G \\ S & C & G & \end{array}$
9	Predicted True $\begin{array}{cccc} 0.975 & 0.025 & 0.000 & S \\ 0.153 & 0.829 & 0.018 & C \\ 0.000 & 0.122 & 0.878 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.922 & 0.077 & 0.001 & S \\ 0.035 & 0.940 & 0.026 & C \\ 0.000 & 0.100 & 0.900 & G \\ S & C & G & \end{array}$
10	Predicted True $\begin{array}{cccc} 0.547 & 0.453 & 0.000 & S \\ 0.024 & 0.976 & 0.000 & C \\ 0.000 & 0.127 & 0.873 & G \\ S & C & G & \end{array}$	Predicted True $\begin{array}{cccc} 0.658 & 0.313 & 0.028 & S \\ 0.006 & 0.989 & 0.006 & C \\ 0.000 & 0.098 & 0.902 & G \\ S & C & G & \end{array}$

As seen in Table 1, adding a few images from an human expert, avoids CNN to behave as an SVM. The advantage is that an human expert need to generate only twenty training images which the network can make a good generalization and correct mistakes generated by the SVM, for example, rows 5 and 6 in Table 1.

From the matrices in Table 2, recall, precision, and F1 score are computed to measure the effectiveness of the algorithm and to compare it with the SPS-SVM. These results are shown in Table 3. There is no entry for SVM in test 8 because, in such an experiment, only two classes were found (missing sky).

The confusion matrices for both techniques are very close. To get a better understanding for each matrix the macro versions of the recall, precision, and f1-score, in Table 3. In Figure 10, the same score to get a better visual understanding of proposal performance is plotted. The proposal overcome the SPS-SVM in almost all the samples (except for the sample seven).

Table 3. The measure of the test for both approaches. Bold letters denote the winner technique

Test	Approach	Recall	Precision	F1 Score
1	SVM	0.8497	0.7962	0.8059
	CNN	**0.8975**	**0.8750**	**0.8801**
2	SVM	0.8789	0.8503	0.8563
	CNN	**0.9113**	**0.8973**	**0.9006**
3	SVM	0.9024	0.8886	0.8914
	CNN	**0.9172**	**0.9121**	**0.9134**
4	SVM	0.9128	0.8964	0.8989
	CNN	**0.9321**	**0.9225**	**0.9238**
5	SVM	0.5127	0.5985	0.5193
	CNN	**0.8345**	**0.6916**	**0.6522**
6	SVM	0.7417	0.6413	0.6314
	CNN	**0.8419**	**0.7063**	**0.6833**
7	SVM	**0.8767**	**0.7751**	**0.7818**
	CNN	0.8470	0.7479	0.7413
8	SVM	-	-	-
	CNN	**0.9231**	**0.9099**	**0.9087**
9	SVM	0.8919	0.8682	0.8741
	CNN	**0.9224**	**0.9161**	**0.9173**
10	SVM	0.9016	0.8786	0.8792
	CNN	**0.9247**	**0.9161**	**0.9173**

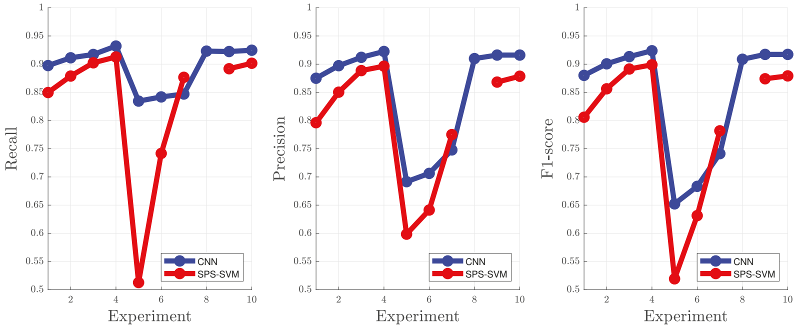


Figure 10. The measure of the test for both approaches.

For both schemes, we find the hyperparameters heuristically guided by the train and test scores obtained from 30 executions episodes. Finally, in this paper, we do not show a run-time comparison because CNN was implemented in TensorFlow-Keras Framework; consequently, it runs over the Graphical Process Unit (GPU). On the other hand, the system SPS-SVM was implemented as a sequential algorithm to be executed on the CPU due to the complexity of its parallelization. CNN has lower run-time than SPS-SVM, but the comparison is not fair until we get a parallelized implementation of SPS-SVM.

7. Conclusions

As can be seen in the previous section, the approach gives good results not only classifying the parts of the environment that are desired to be segmented into classes but also reducing the tedious labor of generating a data set by human hand. As seen on the results image, the proposal can classify with more detail than a SVM or a human using basic image editing tools.

The CNN for pixel classification commonly needs a big data set to train; in this paper, a CNN is pre-trained with the prediction of an SPS–SVM. Then, the SPS–SVM can be considered as a data augmentation process to generate synthetic labeled data.

The approach is fast enough to provide sensitive information in a short time, so a UAV can take decisions with recent information. Future work will focus on improving the classification by adding estimations on the different types of clouds that can be found in the environment and the risk they could represent for a UAV.

Author Contributions: Conceptualization, C.V. and N.A.-D.; methodology, C.V.; software, A.A.G.; validation, G.L-G., C.V. and J.G.-A.; formal analysis, N.A.-D. and A.A.G.; investigation, J.D.R.; writing—original draft preparation, J.G.-A.; writing—review and editing, G.L.-G., J.D.R. and J.G.-A.; visualization, A.A.G. and G.L.-G.; supervision, J.D.R.; project administration, C.V. and N.A.-D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CONACYT México grants numbers CB256769, CB258068, and PN-4107.

Acknowledgments: The authors would like to thank Hydra Technologies de México for providing the data for the development of this work.

Conflicts of Interest: The authors declare no conflict of interest. The founders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Networks
UAV	Unmanned Aerial Vehicle
ReLU	Rectified Linear Unit
SILC	Simple Linear Iterative Clustering
SPS	Superpixel Segmentation
SURF	Speeded Up Robust Features
SVM	Support Vector Machine

References

1. Millbrooke, A. *Aviation History*; Jeppesen: Englewood, CO, USA, 2006.
2. Gupta, S.G.; Ghonge, D.; Jawandhiya, P.M. Review of unmanned aircraft system (UAS). *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)* **2013**, *2*. [CrossRef]
3. Bottyán, Z.; Tuba, Z.; Gyöngyösi, A.Z. Weather Forecasting System for the Unmanned Aircraft Systems (UAS) Missions with the Special Regard to Visibility Prediction, in Hungary. In *Critical Infrastructure Protection Research*; Springer International Publishing: Cham, Switzerland, 2016; pp. 23–34.
4. George, J.J. *Weather Forecasting for Aeronautics*; Academic Press: London, UK, 2014.

5. Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* **2008**, *110*, 346–359. [CrossRef]
6. Wang, J.; Cohen, M.F. Optimized color sampling for robust matting. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 1–8.
7. Ren, X.; Malik, J. Learning a classification model for segmentation. In Proceedings of the Ninth IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; IEEE: Piscataway, NJ, USA, 2003; p. 10.
8. Levin, A.; Lischinski, D.; Weiss, Y. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 228–242. [CrossRef] [PubMed]
9. Zhang, Q.; Xiao, C. Cloud detection of RGB color aerial photographs by progressive refinement scheme. *IEEE Trans. Geosci. Remote. Sens.* **2014**, *52*, 7264–7275. [CrossRef]
10. Peña-Olivares, O.; Villaseñor, C.; Gallegos, A.A.; Gomez-Avila, J.; Arana-Daniel, N. Automatic Environment Classification for Unmanned Aerial Vehicle Using Superpixel Segmentation. In Proceedings of the 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Gudalajara, Mexico, 7–9 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
11. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.
12. Ghonima, M.; Urquhart, B.; Chow, C.; Shields, J.; Cazorla, A.; Kleissl, J. A method for cloud detection and opacity classification based on ground based sky imagery. *Atmos. Meas. Tech. Discuss.* **2012**, *5*, 4535–4569. [CrossRef]
13. Liu, S.; Zhang, L.; Zhang, Z.; Wang, C.; Xiao, B. Automatic cloud detection for all-sky images using superpixel segmentation. *IEEE Geosci. Remote Sens. Lett.* **2014**, *12*, 354–358.
14. Boykov, Y.Y.; Jolly, M.P. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV 2001), Vancouver, BC, Canada, 7–14 July 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 1, pp. 105–112.
15. Xiao, C.; Liu, M.; Xiao, D.; Dong, Z.; Ma, K.L. Fast closed-form matting using a hierarchical data structure. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 49–62. [CrossRef]
16. Black, K.M.; Law, H.; Aldouhki, A.; Deng, J.; Ghani, K.R. Deep learning computer vision algorithm for detecting kidney stone composition. *BJU Int.* **2020**, *125*, 920–924. [CrossRef]
17. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
18. Wang, Z.; Chen, J.; Hoi, S.C. Deep learning for image super-resolution: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [CrossRef]
19. Arabi, S.; Haghighat, A.; Sharma, A. A deep-learning-based computer vision solution for construction vehicle detection. *Comput. Aided Civ. Infrastruct. Eng.* **2020**, *35*, 753–767. [CrossRef]
20. Wu, X.; Sahoo, D.; Hoi, S.C. Recent advances in deep learning for object detection. *Neurocomputing* **2020**, *395*, 39–64.
21. Attari, N.; Ofli, F.; Awad, M.; Lucas, J.; Chawla, S. Nazr-cnn: Fine-grained classification of uav imagery for damage assessment. In Proceedings of the 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Tokyo, Japan, 19–21 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 50–59.
22. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995; pp. 276–279.
23. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
24. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv* **2013**, arXiv:1312.6229.
25. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
26. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.

27. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
28. Eigen, D.; Fergus, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2650–2658.
29. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
30. Muller, K.R.; Mika, S.; Ratsch, G.; Tsuda, K.; Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **2001**, *12*, 181–201. [CrossRef]
31. Haykin, S. *Neural Networks: A Comprehensive Foundation*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 1994.
32. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef]
33. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
34. Kruizinga, P.; Petkov, N. Nonlinear operator for oriented texture. *IEEE Trans. Image Process.* **1999**, *8*, 1395–1407. [CrossRef]
35. Grigorescu, S.E.; Petkov, N.; Kruizinga, P. Comparison of texture features based on Gabor filters. *IEEE Trans. Image Process.* **2002**, *11*, 1160–1167. [CrossRef] [PubMed]
36. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

applied sciences

MDPI

Article

Graphs Regularized Robust Matrix Factorization and Its Application on Student Grade Prediction

Yupei Zhang, Yue Yun, Huan Dai, Jiaqi Cui and Xuequn Shang *

School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, Shaanxi, China; ypzhaang@nwpu.edu.cn (Y.Z.); yundayue@mail.nwpu.edu.cn (Y.Y.); daihuan@mail.nwpu.edu.cn (H.D.); cuijiaqi@nwpu.edu.cn (J.C.)

* Correspondence: shang@nwpu.edu.cn

Received: 9 January 2020; Accepted: 21 February 2020; Published: 4 March 2020

Abstract: Student grade prediction (SGP) is an important educational problem for designing personalized strategies of teaching and learning. Many studies adopt the technique of matrix factorization (MF). However, their methods often focus on the grade records regardless of the side information, such as backgrounds and relationships. To this end, in this paper, we propose a new MF method, called graph regularized robust matrix factorization (GRMF), based on the recent robust MF version. GRMF integrates two side graphs built on the side data of students and courses into the objective of robust low-rank MF. As a result, the learned features of students and courses can grasp more priors from educational situations to achieve higher grade prediction results. The resulting objective problem can be effectively optimized by the Majorization Minimization (MM) algorithm. In addition, GRMF not only can yield the specific features for the education domain but can also deal with the case of missing, noisy, and corruptive data. To verify our method, we test GRMF on two public data sets for rating prediction and image recovery. Finally, we apply GRMF to educational data from our university, which is composed of 1325 students and 832 courses. The extensive experimental results manifestly show that GRMF is robust to various data problem and achieves more effective features in comparison with other methods. Moreover, GRMF also delivers higher prediction accuracy than other methods on our educational data set. This technique can facilitate personalized teaching and learning in higher education.

Keywords: robust matrix factorization; student grade prediction; educational data mining; side information graph; personal teaching and learning

1. Introduction

In high school education, student grade prediction (SGP) can make great sense for aiding all stakeholders in the education process. For students, SGP can help them to choose suitable courses or exercises for increasing their knowledge, and even to make their pre-plans for academic periods. For instructors, SGP can help them to adjust learning materials and teaching programs based on student ability, and to find the students that are at risk of disqualification in course progress. For educational managers, SGP can help them to check the curriculum program and to arrange the courses in a scientific order. All stakeholders of the educational process could have a better self-plan to improve education outcomes and then have a higher graduation rate. SGP is an important problem for scientific education in STEM (Science, Technology, Engineering, Mathematics), referred to in the work of G. Shannon et al. [1].

Appl. Sci. **2020**, *10*, 1755; doi:10.3390/app10051755 www.mdpi.com/journal/applsci

Student grade prediction aims to predict the final score/grade of course enrolled by a target student in the next academic term. SGP provides a useful reference to evaluate educational outputs in advance and is thus significant necessary for various tasks towards personalized education, such as ensuring on-time graduation [2] and improving learning grade [3,4]. Over the past years, many studies have paid attention to SGP and have already developed many methods [5].

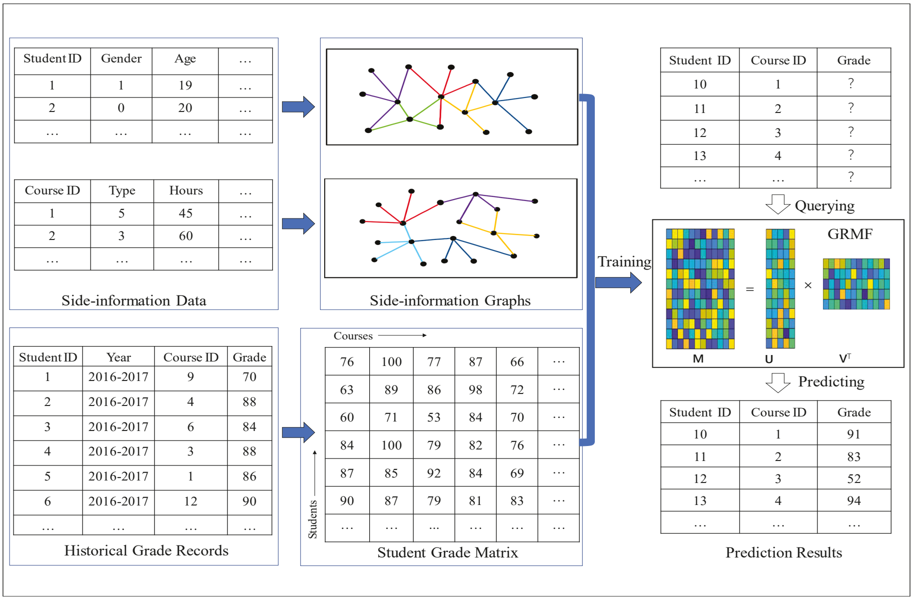
Existing methods can be principally divided into three categories depending on their formulation, as follows: (1) Classification problem. SGP is recast as labeling the target student with the predefined grade tags and was solved by classification models, such as decision tree [6], logic regression [7,8] and support vector machine [9,10]. (2) Regression problem. By taking the grade as the response variable, SGP is rewritten into assigning scores following the features of student or course, such as linear regression [5,11,12], neural network [13–15] and random forest [9]. (3) Matrix completion. Since grade records can be poured into a matrix, SGP is also formulated as predicting the missing values of the student-course matrix with each element being a course grade [16]. This formulation is usually solved by the popular method of matrix factorization and has been extensively studied, leading to many effective approaches [17,18]. In particular, based on the same dataset, Thai-Nghe et al. compared matrix competition with traditional regression methods such as logistic/linear regression and the experimental results show that matrix competition can improve prediction results [19].

MF based methods aim to learn the latent features of student and course from the given grade data and then uses these features for SGP [20]. Here, we review the related works that using MF techniques. Traditional MF was employed to implicitly encode "slip rate" (the probability that the student knows how to solve a question but makes a mistake) and the "guess rate" (the probability that the student does not know how to solve a question but guesses correctly) of the student in an examination, resulting in an excellent performance on the educational data set of KDD (Knowledge Discovery and Data Mining) Cup 2010 [21]. In References [22,23], Non-negative Matrix Factorization (NMF) was used to integrate the nonnegativity of student grade. Tensor factorization (TF) was exploited to take the temporal effects into account in Reference [24], due to the improvement of the ability of students. Since grade matrix is implicitly low rank, low-rank matrix factorization (LRMF) was investigated in data sets from the online learning platform in the work of Lorenzen et al. [25]. But the existing MF based methods often suffer from the issues of missing data, corrupted data, and data noise. Especially, they fail to consider the side information which is included in the other handy educational data, such as background data and daily behavior data in school.

Since the L_2-norm based reconstruction is sensitive to outliers and data corruptions, Lin et al. proposes to use L_1-norm instead of L_2-norm to enhance the robustness [26–28]. Besides, we often have massively available side information data in real-world applications. Rao et al. proposes a method of graph regularized alternating least squares (GRALS) to integrated two graphs from the side information data of movies and viewers [29]. More specifically, in the real context of high education, the data set usually has the following properties: (1) The grade matrix is heavily lost for course selection and corrupted by some human factors. (2) The students with similar backgrounds are likely to have similar performance in a course [30]. For example, two students both have more exercises in computer programming, and then they may both obtain a perfect grade at their course of *C language* with a high probability. (3) The courses with similar knowledge tend to give rise to a similar grade for a student. For instance, *C Language* is similar with *Data Structure* while *C Language* is not similar with *History*, thus student who is good at *C Language* is likely to have good performance in *Data Structure* but not necessarily *History*.

To this end, we put forth a novel MF method, called double graph regularized robust matrix factorization (GRMF), following by applying GRMF for SGP as shown in Figure 1. GRMF not only uses the robust loss function from RMF-MM but also integrates two side information graphs constructed using the background data of students and courses. The MM algorithm can effectively solve the resulting optimization problem. Two-folds contributions of our paper are summarized as follows:

- We propose a new model of matrix factorization, dubbed Graph regularized Robust Matrix Factorization (GRMF), by considering both the robustness to data pollution and the side information from background descriptions.
- We design a workflow by applying GRMF on the problem of SGP, shown in Figure 1, where the real-world data set is collected from our university.



Student ID	Gender	Age	…
1	1	19	…
2	0	20	…
…	…	…	…

Course ID	Type	Hours	…
1	5	45	…
2	3	60	…
…	…	…	…

Side-information Data

Side-information Graphs

Student ID	Year	Course ID	Grade
1	2016-2017	9	70
2	2016-2017	4	88
3	2016-2017	6	84
4	2016-2017	3	88
5	2016-2017	1	86
6	2016-2017	12	90
…	…	…	…

Historical Grade Records

76	100	77	87	66	…
63	89	86	98	72	…
60	71	53	84	70	…
84	100	79	82	76	…
87	85	92	84	69	…
90	87	79	81	83	…
…	…	...	…	…	…

Student Grade Matrix

Student ID	Course ID	Grade
10	1	?
11	2	?
12	3	?
13	4	?
…	…	?

Student ID	Course ID	Grade
10	1	91
11	2	83
12	3	52
13	4	94
…	…	…

Prediction Results

Figure 1. The proposed workflow of student grade prediction using GRMF.

The rest of this paper is organized as follows—in Section 2, we formulate the problem of SGP, followed by brief reviewing the MF technique. We present GRMF in Section 3 and the GRMF algorithm in Section 4. Section 5 shows the experimental results on movie rate prediction, image recovery, and SGP. Section 6 finally concludes this paper.

2. Ralated Works

In this section, we formulate the problem of SGP in the form of mathematics, followed by introducing the promising technique of matrix factorization.

2.1. Student Grade Prediction (SGP)

In current higher education in university, the teachers provide a "one-size-fits-all" curriculum, while the students enroll many courses to obtain academic credit. To graduate on time, the student expects to know which course he/she can pass with high score/grade, while the teacher expects to know which student has a risk of failure in his/her course. Hence, the problem of predicting the student grade at a course is significant to improve the educational outcomes.

Generally speaking, the grade of one student at a target course can be inferred by his/her learning records, including historical grades in enrolled courses, academic behaviors and his/her background [31,32]. In this paper, we make the following assumption—the grade can be determined by the latent features of student and course, where those features can be derived from the data of students and courses. We explicitly define the task of SGP as follows:

Problem 1 (Student Grade Prediction): Let $g(s,c)$ be the grade of student s at course c. Denote by $\mathbf{u}_s$ the feature of student s and $\mathbf{v}_c$ the feature of course c. Given the grade matrix $\mathbf{M}$, SGP aims to seek the mapping $\mathcal{H}(\mathbf{u}_s, \mathbf{v}_c)$, such that $g(s,c) = \mathcal{H}(\mathbf{u}_s, \mathbf{v}_c)$ for all grades in $\mathbf{M}$.

To solve Problem 1, we should extract $\mathbf{u}$ and $\mathbf{v}$ and design a mapping using the given data matrix $\mathbf{M}$. Most research designs or learns the features by using the background information [33,34], such as student age and credit time, or the student grades on all finished courses. Since both of them are helpful, in this paper, we combine both information for SGP through developing the MF [26].

2.2. Matrix Factorization

Letting $\mathbf{M} \in \mathbb{R}^{m\,\times\,n}$ be the given matrix, MF aims to seek two latent feature matrices $\mathbf{U} \in \mathbb{R}^{m\,\times\,k}$ and $\mathbf{V} \in \mathbb{R}^{m\,\times\,k}$ to approximate $\mathbf{M}$. The traditional MF can be written as:

$$\min_{\mathbf{U},\mathbf{V}} ||\mathbf{M} \ - \ \mathbf{U}\mathbf{V}^T||_F, \tag{1}$$

where k is the number of latent features predefined in $\mathbf{U}$ and $\mathbf{V}$, and $||\cdot||_F$ is the Frobenius norm. Optimization problem (1) can be solved by various algorithms, such as Majorization Minimization (MM) [35], alternating the direction of the method of multipliers (ADMM) [36], simulated annealing (SA) [37]. Besides, many variants of MF have been proposed, including LRMF [25], NMF [22] and TF [24].

To enhance the robustness, robust matrix factorization via majorization minimization (RMF-MM) employs L_1-norm instead of L_F-norm as the reconstruction term [26]. The objective problem of RMF-MM is:

$$\min_{\mathbf{U},\mathbf{V}} ||\mathbf{W} \odot \left(\mathbf{M} - \mathbf{U}\mathbf{V}^T\right)||_1 \ + \ \frac{\lambda}{2}||\mathbf{U}||_F^2 \ + \ \frac{\lambda}{2}||\mathbf{V}||_F^2\,, \tag{2}$$

where $||\cdot||_1$ is L_1-norm of matrix, $\lambda > 0$ is a regularization parameter and $\mathbf{W}$ is defined as follows:

$$\mathbf{W}_{ij} = \begin{cases} 0, & \text{the value of } \mathbf{M}_{ij} \text{ is missing} \\ 1, & \qquad\qquad\qquad \mathit{otherwise}. \end{cases} \tag{3}$$

The problem above can be effectively optimized by MM algorithm. The results in the experiments by Lin et al. show that RMF-MM is robust to high missing rate or severe data corruption [26].

Since RMF-MM can effectively learn the features from noisy data and then uses the features for prediction, we reformulate the Problem 1 for employing this novel technique, as follows:

Problem 2. (SGP-MF): Given a student grade matrix $\mathbf{M}$, SGP-MF aims to extract $\mathbf{U}$ for students and $\mathbf{V}$ for courses such that $\mathbf{M} = \mathbf{U}\mathbf{V}$. Then the target grade is predicted by

$$g(s,c) = \mathbf{M}_{s,c} = \mathbf{u}_s^T \mathbf{v}_c, \tag{4}$$

where $g(s,c)$ is the grade of student s on course c, $\mathbf{u}_s$ is the s-th row of $\mathbf{U}$ and $\mathbf{v}_c$ is the c-th row of $\mathbf{V}$. And $\mathbf{M}_{s,c}$ is the element of s-th row, c-th column of matrix $\mathbf{M}$.

The reason we consider the Formula (4) is the fact that a student enrolls on a course and obtains a grade. This fact motivates us to obtain the student's features and course's features, given the grade matrix. In this paper, we consider this problem using the matrix factorization (MF) method. As in Formula (4), each grade $M_{s,c}$ is made by $\mathbf{u}_s^T\mathbf{v}_c$ to obtain the latent features

However, RMF-MM fails to consider the side-information data that is often available. The method of graph matrix factorization (GMF) is an approach to integrate the neighborhood structure of $\mathbf{M}$, but it does not work for matrix completion [38]. Based on GMF, we here solve the SGP by combining two side information graphs with RMF-MM.

3. Double Graph Regularized Robust Matrix Factorization

In this section, we present our motivation for considering side information data in SGP and encode them into two graphs, followed by our objective problem and its detail optimization with MM.

3.1. Motivation

In real-world education, various related information can be obtained from the student, such as background, daily life, and student behaviors, as well as course. These side information data contain the relationships among students and courses that can be used for enhancing the prediction performance. Hence, we in this paper propose to encode them in two graphs, followed by integrating them into RMF.

More specifically, we list some *observations*: (1) The family background, such as the economic situation and educational level of their parents, influences the scope of student knowledge [39]. (2) The background of students, such as majors and ages, may affect their habits of thinking and learning. (3) The related course contains much overlapping knowledge or similar skills. (4) Courses taught by an identical teacher are similar in the style of teaching and testing [40].

From the above observations, we have the follows: On the one hand, it is believed that students with a similar background can obtain similar performance. On the other hand, two similar courses tend to have similar grade distribution.

3.2. Side Information Graph

Considering the row/column vectors of $\mathbf{M}$ as data points, each row vector of $\mathbf{U}/\mathbf{V}$ is the low-rank representation of the corresponding row/column in $\mathbf{M}$. Note that each row in both $\mathbf{M}$ and $\mathbf{U}$ corresponds to a student, while each column in both $\mathbf{M}$ and $\mathbf{V}^T$ corresponds to a course. Besides, we have side information feature matrixes from students and courses, denoted by $\mathbf{S}_u$ and $\mathbf{S}_v$. Following above, if two students/courses are close in terms of $\mathbf{S}_u$ /$\mathbf{S}_v$, then the corresponding rows of $\mathbf{U}/\mathbf{V}$ are also close to each other [41,42].

In order to simultaneously integrate the side information of students and courses, we knit two similarity graphs using $\mathbf{S}_u$ and $\mathbf{S}_v$ instead of using $\mathbf{M}$ [38,43,44]. That is the reason that the graphs here are called side information graph. The method of building graph is as follows. Denote by $\mathbf{Q} = \{\mathbf{S}, \mathbf{E}|\mathbf{G}\}$ the side information graph, where $\mathbf{S}$ includes all data points from students or courses, $\mathbf{E}$ is the set of edges, and $\mathbf{G}$ contains all weights on all edges. $\mathbf{G}$ is constructed by:

$$\mathbf{G}_{ij} = \begin{cases} e^{-\frac{||\mathbf{s}_i \; - \; \mathbf{s}_j \; ||^2}{\sigma}} & , \quad \mathbf{s}_i \in N_k\left\{\mathbf{s}_j\right\} \text{ or } \mathbf{s}_j \in N_k\left\{\mathbf{s}_i\right\} \\ 0 & , \quad \text{otherwise} \end{cases}, \tag{5}$$

where $\mathbf{s}_i$ is corresponding to the data point in $\mathbf{S}_u$ or $\mathbf{S}_v$, σ is the kernel parameter and $N_k\{\mathbf{x}\}$ indicates the set of k neighbors to sample $\mathbf{x}$. The details can be found in the literature [41].

Since the similarity relationships encoded in the side information graphs are constructive for learning the latent features, we hope to preserve them in $\mathbf{U}$ and $\mathbf{V}$. Taking $\mathbf{U}$ for example, we, as usual, employ the following objective [41]:

$$R_1 = \frac{1}{2}\sum_{i,j}\mathbf{G}_{i,j}||\mathbf{u}_i - \mathbf{u}_j||_2^2 = tr\left(\mathbf{U}^T\mathbf{H}_u\mathbf{U}\right), \tag{6}$$

where $tr(\cdot)$ denotes the trace of a matrix, $\mathbf{u}_i$ is the row of $\mathbf{U}$ and $\mathbf{H}_u = \mathbf{D} - \mathbf{G}$, $\mathbf{D}_{ii} = \sum_j \mathbf{G}_{i,j}$. Similarly, we can knit the side information graph of course and then obtain two Laplacian regularization terms.

3.3. The Objective Problem of GRMF

With the idea of integrating the side information, we combine the objective of RMF-MM and the two Laplacian regularizations, as follows:

$$\begin{aligned}\min_{\mathbf{U},\mathbf{V}} ||\mathbf{W} \odot (\mathbf{M} - \mathbf{U}\mathbf{V}^{\mathbf{T}})||_1 + \frac{\lambda}{2} (||\mathbf{U}||_F + ||\mathbf{V}||_F) + \\ \frac{\alpha}{2} \left(tr \left(\mathbf{U}^T \mathbf{H}_u \mathbf{U} \right) + tr \left(\mathbf{V}^T \mathbf{H}_v \mathbf{V} \right) \right),\end{aligned} \tag{7}$$

where $\lambda > 0, \alpha \geq 0$ are two trade-off parameters, and $\mathbf{H}_u/\mathbf{H}_v$ is defined in the above section. From (7), we can believe that GRMF can reach a better performance than RMF-MM ,since GRMF degenerates into RMF-MM when α is zero.

The main difference between GRMF and RMF-MM lies in the graph Laplacian regularizers of (6), where GRMF integrates more data priors. While GRALS uses L_2-norm for data fidelity [29]. GRMF proposes to adopt L_1-norm and thus is more robust to data noise and pollution.

The SGP problem is first described as a machine learning problem, shown in Problem 1. We assume the grade is determined by the student's latent features and the course's latent features. This assumption is general. The problem is then reformulated by matrix factorization (MF), since we plan to adopt MF to learn the latent features. In order to consider the noise in the given grade matrix, we reformulated the objective of MF by L_1 normal, because the noise is considered from the grade temper, slipping, and so forth. Finally, for better prediction result, we consider the relationship of students and the relationship of courses in our robust MF model though two graph regularization items. Our objective is thus shown in Equation (7).

4. GRMF Algorithm

In this section, we use a majorization-minimization algorithm to solve problem (7). Suppose that we already have obtained $(\mathbf{U}_k, \mathbf{V}_k)$ after the k-th iterations. We split $(\mathbf{U}, \mathbf{V})$ as the sum of $(\mathbf{U}_k, \mathbf{V}_k)$ and the unknown residue $(\Delta\mathbf{U}_k, \Delta\mathbf{V}_k)$:

$$(\mathbf{U}_{k+1}, \mathbf{V}_{k+1}) = (\mathbf{U}_k, \mathbf{U}_k) + (\mathbf{\Delta U}_k, \mathbf{\Delta V}_k). \tag{8}$$

The task can now be finding the small increment $(\Delta\mathbf{U}_k, \mathbf{\Delta V}_k)$ in the k-th iteration such that the objective function keeps decreasing. To seek the best $(\Delta\mathbf{U}_k, \mathbf{\Delta V}_k)$, we employ the linearized Direction Method with Parallel Splitting and Adaptive Penalty (LADMPSAP) [45]. We made the detailed procedure of this optimization in Appendix A. We summarize the main flow of GRMF to make the paper self-contained in Algorithm 1, shown as below:

Algorithm 1 Graph regularized Robust Matrix Factorization (GRMF) by Majorization Minimization

```
Input: $\mathbf{M} \in \mathbb{R}^{n\times m}$, $\alpha$, and $\lambda$
Output: U and V
Method:
  Initialize $\mathbf{U}_0$ and $\mathbf{V}_0$ with using SVD on M;
  $\Delta\mathbf{U}^0 = \Delta\mathbf{V}^0 = 0$; $\varepsilon_1 = \varepsilon_2 = 1e - 6$.
  While not converged when we arrived $(\mathbf{U}_k, \mathbf{V}_k)$, do
      Let $t = 1$;
      While not converged, do
          Update $\Delta\mathbf{U}^t$ and $\Delta\mathbf{V}^t$ via LADMPSAP;
          $t = t + 1$;
      End while
      $(\Delta\mathbf{U}_k, \Delta\mathbf{V}_k) = (\Delta\mathbf{U}_t, \Delta\mathbf{V}_t)$;
      Update U and V in parallel:
          $\mathbf{U}_{k+1} = \mathbf{U}_k + \Delta\mathbf{U}_k$;
          $\mathbf{V}_{k+1} = \mathbf{V}_k + \Delta\mathbf{V}_k$;
      Check the convergence coditions, if
          $\mathbf{V}_{k+1} - \mathbf{V}_k < \varepsilon_1$ and $\mathbf{U}_{k+1} - \mathbf{U}_k < \varepsilon_2$;
  End while.
```

5. Experimental Results

In order to evaluate the performance of GRMF, we conducted the following experiments: (1) testing GRMF, RMF-MM and on MOVIELENS 100k datasets and a public image data; (2) comparing GRMF with several fashion methods for student grade prediction, including RMF-MM [26], GRALS [29], MF [46], NMF [22], PMF [47], KNN(k-Nearest Neighbor) [48] and column mean [49] using the real educational dataset from our university. Note that MF is the standard matrix factorization solved with gradient descent; column-mean is the mean scores of historical grades of target course; and for KNN-mean, we obtained the k neighbor students and then computed the grade mean. The code and data sets are available on our website, https://github.com/ypzhaang/student-performance-prediction.

5.1. Evaluation Metric

Three metrics are used for evaluating the results: Root Mean Squared Error (RMSE), L_1-norm Error (Err1) [26], PSNR (Peak Signal to Noise Ratio) and Acc (Accuracy rate). Especially, in our paper, Acc is computed as follows:

$$Acc = \frac{\sum_{i=1}^{n} \Delta g_i}{n}, \tag{9}$$

where

$$\Delta g = \begin{cases} 1, & |(g_{re} - g)| \geq 0.5 \\ 0, & |(g_{re} - g)| < 0.5' \end{cases} \tag{10}$$

in which g_{re} is the predicted grade while g is the true grade and n is the number of grades.

5.2. Test on a Toy Data from Movie Dataset

MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. These data sets consist of 100,000 ratings (1–5) from 943 users on 1682 movies, background information from users (e.g., age, occupation, and zip code) and movies (e.g., title, release date, and genre). Besides, users who have less than 20 ratings or do not have completed demographic information were removed. In this test experiment, we draw out a toy data set from MovieLens to probe the effectiveness, convergence, and parameter effects of GRMF. And in the toy data set, the user ids are less than 200, and the movie ids are less than 300.

5.2.1. Rating Prediction and Algorithm Convergence

We divided the toy data set into a training set and test set by random sampling. To evaluate the small toy data, we employed a five-fold cross validation that trains models on four-fold samples and tests on the remaining samples. Whereby we constructed two five-nearest neighborhood graphs from the background data of both users and movies. We chose suitable parameters for achieving best performance using all the mentioned methods. Note that the optimal parameters of GRALS were selected in Reference [29].

Table 1 shows the prediction results from using four methods on the toy data set. It is easy to observe that: (1) MF is better than RMF-MM and GRALS in terms of RMSE, but worse than the two latter compared to Err1. (2) RMF-MM has better performance on Err1 than GRALS, which is more robust to evaluate. (3) Overall, our method delivers the best results using either RMSE or Err1. All the above says that GRMF can benefit from the side information data to enhance rate prediction performance.

In addition, Figure 2 displays the convergence proceeding of GRMF on the toy data. As is shown, GRMF can converge to stable Err1 after about 16 iterations. With more observations on other data sets, Algorithm 1 can have a fast convergence and arrive at an effective solution.

Table 1. Err1 and RMSE on toy dataset.

	Err1	**RMSE**
GRMF	**0.735**	**0.957**
MF	1.549	1.007
GRALS	0.770	1.008
RMF-MM	0.776	1.104

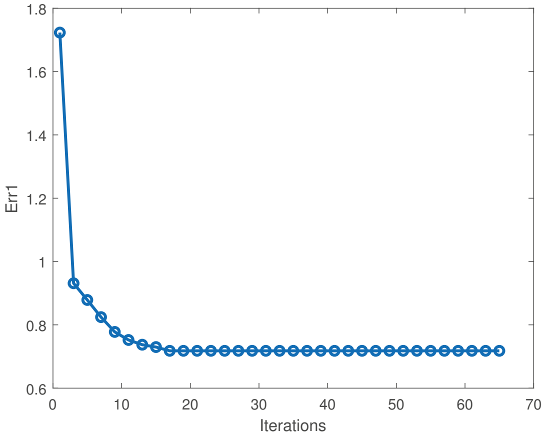
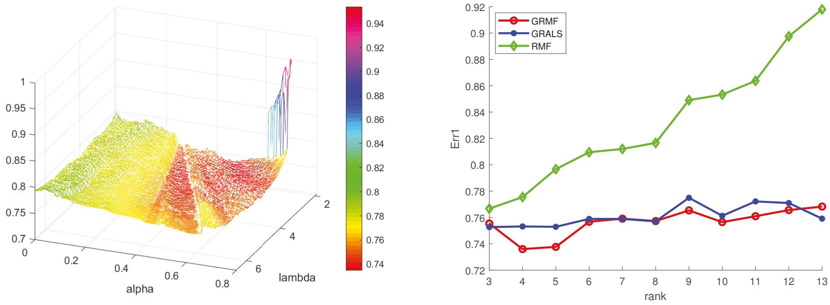


Figure 2. The value of Err1 versus iterations of Graph Regularized Robust Matrix Factorization (GRMF) on toy dataset.

5.2.2. The Effects of Parameters on Rating Prediction

We have the graph regularization parameter α, the regularization parameter λ and the rank of factorized matrices k in the objective (7) of GRMF. We here discuss the effects of these three parameters on the prediction performance utilizing the above toy data set on our prediction experiment.

The two parameters of α and λ vary in wide ranges as is shown in Figure 3b. Figure 3b shows the 3D curve of Err1 created under the effects of α and λ. From the curve, we observe that there is a broad range of parameter pairs that can be available for producing decent prediction results. Besides, we also probe the effect of the parameter k, shown in Figure 3a. The results show that GRMF has the most stable performance under varying k, while the RMF-MM has the worst performance.



(a) λ and α

(b) k

Figure 3. The effects of the parameters of GRMF on the Err1.

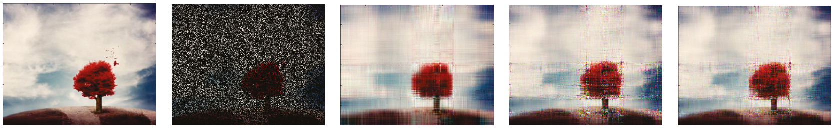
5.3. Evaluation on Image Data Set

The problem of image recovery is often formulated as matrix completion. Since the top singular values dominate the main information, most of the images could be regarded as a low-rank matrix. Hence, we apply the proposed method to recover the image from its noisy version. This test aims to recast the experiment conducted in the work of Lin et al. [26]. Concretely, we pollute the images (https://sites.google.com/site/zjuyaohu/) with Gaussian noise or salt-and-pepper noise, then recover the images from the noisy version in comparison with the methods of RMF-MM and GRALS.

5.3.1. Gaussian Noise

We added Gaussian noise with the variance being 1 and mean being 0 to g percent of the observed pixels, where g is the corruption ratio. Figure 4b shows the example image which was corrupted with Gaussian noise. g was varied in the range of [45, 90] to observe the performance in various situations. We ran the three methods to recover the corrupted image in Figure 4b, where the side-information data consists of the rows and columns of the corrupted image. Figure 5 shows the PNSRs (Peak Signal to Noise Ratios) from the three compared methods. From the curves, GRMF consistently achieves the highest PSNRs on all test cases. When the corruption ratio increases, GRMF delivers a much better result than RMF-MM. Note that GRALS has a weak performance because its reconstruction term is very sensitive to data pollutions.

Figure 4c–e depicts the resultant images from the case of g = 80, using the three methods. As is clear, our GRMF produces the best visualization, while the other methods suffer a few horizontal or vertical lines.



(a) Original (b) Corrupted (c) GRALS (d) RMF-MM (e) GRMF

Original 11.60dB 23.40dB 24.01dB 28.75dB

Figure 4. The PNSRs of Image recovery with Gaussian noise.

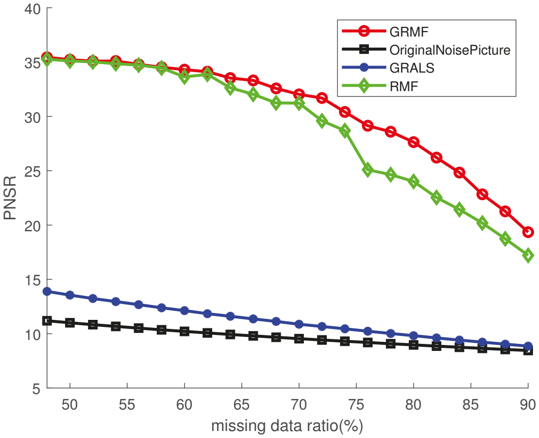


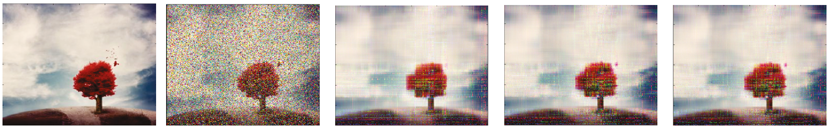
Figure 5. Evaluation of image recovery with Gaussian noise in term of PSNR. The black line that marked with "Corrupted" means the PNSR of the corrupted images.

5.3.2. Salt-And-Pepper Noise

We added the salt-and-pepper noise with noise density varying from 0.05 to 0.65 with a step of 0.05 to image and obtained the corrupted image, like Figure 6b. Then, we ran the three compared methods for denoising the corrupted image where the side-information consists of the rows and the columns of the corrupted image. Figure 7 shows the results of image denoising by GRALS, RMF-MM, and GRMF. From Figure 7 it is clear that GRMF delivers the best performance on PSNR when the noise density is less than 0.4 but drops down if the noise density is greater than 0.5, where the other two methods obtained worse results. The reason is that most pixels of the image are corrupted so that the graphs are difficult to obtain well in a noisy situation. In addition, Figure 6 shows the resultant images when the noise density is 0.4, where our method touches the highest PNSR of 22.88 dB.

5.4. Application on Educational Data Set

The data were collected from the school of Computer Science, Northwestern Polytechnical University (NPU), across students who joined in the past five years, that is, from 2013 to 2017. We collected all the score/grade recorders before the fall of 2017, together with the side information.



(a) Original (b) Corrupted (c) GRALS (d) RMF-MM (e) GRMF

Original 13.58db 19.36dB 20.86dB 22.88dB

Figure 6. The Results of Image recovery with salt-and-pepper noise.

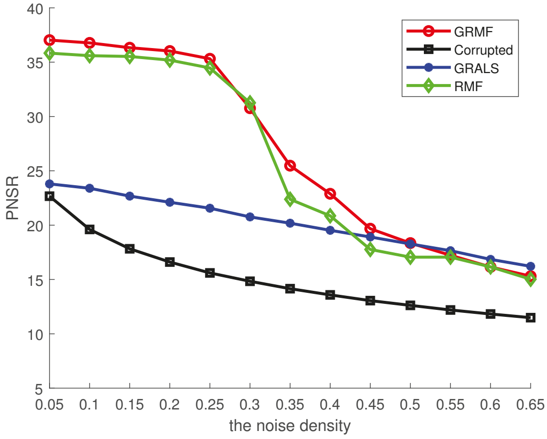


Figure 7. Evaluation of image recovery with Salt-and-pepper noise.

More specifically, our dataset contains the grades, the side data of student and the side data of courses, respectively denoted by NPU-G, NPU-S, and NPU-C for short. NPU-G is composed of 1325 × 832 grade records from 1325 students at 832 courses. NPU-S contains 25 description features of 1325 students, such as ages, gender, and department. NPU-C includes 18 description features of 882 academic/elective courses, such as hours, type, and course credit. In addition, at least 15 students enrolled and obtained grades in each course, and students starting university in 2013 and 2014 have already completed their program.

SGP in our educational data set has the following challenges: (1) Data sparsity. There are 832 courses in NPU-G, but each student is only required to enroll in a small number of courses, i.e., about 85 courses in our data. (2) data corruption. Many subjective factors affect the final grade, e.g., subjective questions. (3) missing data. A few students do not attend the final exam, and thus give an empty grade in the information system. All this noisy information makes our problem very challenging.

5.4.1. Educational Data Preprocessing

For NPU-G, we removed the students who had lost most of the data records or had taken less than 4 courses, and then removed those courses that were taken by less than 15 students, followed by deleting the secondary courses to ensure a single record per student. Finally, we formulated the remaining records from 882 students and 82 courses into the matrix **M**, ordered by scholar terms. In addition, we transformed the scores into grade 1–6 using the following piecewise function:

$$y = \begin{cases} 1 & 0 < x < 60; \\ 2 & 60 \leq x < 70; \\ 3 & 70 \leq x < 80; \\ 4 & 80 \leq x < 90; \\ 5 & 90 \leq x < 100; \\ 6 & x = 100. \end{cases} \tag{11}$$

where x is the score in the grade record while y is its corresponding grade.

Responding to $\mathbf{M} \in \mathbb{R}^{882\times15}$, we also removed the student and the course from NPU-S and NPU-C. In all collected side descriptions, we selected 15 and 12 important features for NPU-S and NPU-C, respectively, using teaching experience. Finally, we formulated them into matrices $\mathbf{S}_u \in \mathbb{R}^{882\times15}$ for students and $\mathbf{S}_v \in \mathbb{R}^{82\times12}$ for courses.

5.4.2. Implementation Details

We here predict the student grade for each academic term, because of the usual stages at the university. Hence, we used historical records as a training set to predict the grade in the next term. That is, our model was trained on the records from the 1-th to the $(t-1)$-th terms and was tested on the t-th terms. Concretely, in the SGP tasks for the t-th term, we built k-nearest neighborhood graph $\mathbf{G}_u/\mathbf{G}_v$ on the side data of students and courses $\mathbf{S}_u/\mathbf{S}_v$, respectively. Then, we learned the latent features of student and course on training data using our model, followed by computing the evaluation matrix Err1 and Acc. We conduct this experiment on six data splits, where the sizes of training sets and test sets are listed in Table 2.

Table 2. The size of training sets and test sets.

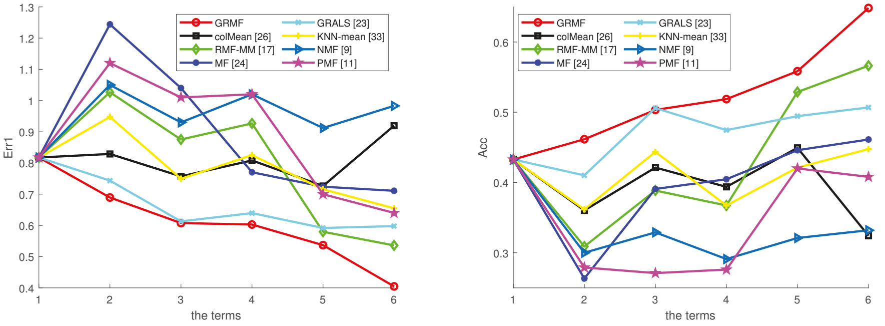
Academic Term	Training Set	Test Set
1	17,425	3189
2	20,779	2043
3	22,821	2692
4	25,506	1473
5	26,949	2063
6	29,045	219

In order to compare with other methods for SGP, we conducted an experiment using MF (S. Rendle, 2010 [46]), NMF (C.S. Hwang, et al., 2015 [22]), PMF (B. Jiang, et al. [47]), KNN (N.C. Wong, et al., 2019 [48]) and column mean (M. Sweeney, et al. [49]). Besides, we also implement SGP using RMF-MM (Z. Lin, et al., 2018 [26]) and GRALS(N. Rao, et al. [29]). For each method, we selected the optimal one from the wide range suggested by their related reports.

5.4.3. Experimental Result and Discussion

Figure 8a and Figure 8b show the prediction results from varying all six terms by various methods in terms of Err1 and Acc. From the curves and comparisons, we observe that: (1) as the semester progresses, the prediction decreases in Err1 and increases in accuracy rate; (2) both GRMF and GRALS are better than other comparable methods; (3) GRMF is not only better than RMF but also outperforms GRALS. (4) the prediction performance of colMean can be regarded as a base performance of SGP. Both GRMF and GRALS can perform better than colMean over all the terms while other methods, including the RMF-MM performance are worse than colmean in most cases.

From these observations, we derive the following conclusions: (1) As the semester progresses, we obtain more information about the student/course which is reflected in the better prediction performance. (2) Side information data of student and course is helpful for SGP. (3) The combination of the side information and the robust L_1 regularizers in our methods GRMF improves the prediction performance effectively. (4) The methods using side information do perform well but other comparable methods cannot handle the prediction task well in the real education context due to the complex problem of real educational data. (5) Our proposed method outperforms traditional classification methods and regression methods. (6) The proposed method GRMF can achieve the accuracy of 65.4% in the sixth term, which is more interesting than the other methods.



(a) Err1

(b) Acc

Figure 8. The effects of the parameters of GRMF on the Err1 and Acc.

6. Discussion and Conclusions

In this paper, we solve the student grade prediction (SGP) problem by proposing a novel matrix factorization method that is dubbed GRMF. GRMF integrates the side information with the robust objective function of matrix factorization, which can be effectively solved by the MM optimization algorithm. The extensive experiments are conducted on movie data, image data, and our education data for testing the performance on rate prediction, image recovery, and SGP. The evaluation results by the used matrices show that GRMF can deliver a better performance than all compared methods. In SGP, GRMF can achieve the highest accuracy of about 65.4%. However, it is still weak in our challenging data. We will improve GRMF and try other fashionable methods to pursue a higher accuracy, while boosting a personalized education.

In addition, a function *f* that maps from **U** and **V** to the grade matrix **G** could be used to achieve a better prediction model, due to the gap between the predicted grade and the real grade. That is because the noise is often caused by accidental events, like exam slipping and guessing. Our study has this limitation on considering this noise in grade prediction. Adding this map *f* may help to obtain more accurate results in the real-world environment. We leave this study for future work.

Author Contributions: Conceptualization, Y.Z. and Y.Y.; Data curation, Y.Y. and J.C.; Formal analysis, Y.Z.; Funding acquisition, Y.Z. and X.S.; Investigation, Y.Y. and H.D.; Methodology, Y.Z.; Resources, J.C. and X.S.; Software, Y.Z. and Y.Y.; Supervision, X.S.; Validation, H.D. and J.C.; Writing—original draft, Y.Z. and Y.Y.; Writing—review & editing, H.D., J.C. and X.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grants No. 61802313, 61772426, U1811262) and the Fundamental Research Funds for Central Universities (Grant No. G2018KY0301).

Acknowledgments: We thank the editors and any reviewers for their helpful comments.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Objective Minimization

Suppose that we already have obtained $(\mathbf{U}_k, \mathbf{V}_k)$ after the kth iterations. We split $(\mathbf{U}, \mathbf{V})$ as the sum of $(\mathbf{U}_k, \mathbf{U}_k)$ and the unknown residue $(\Delta\mathbf{U}, \Delta\mathbf{V})$.

$$(\mathbf{U}, \mathbf{V}) = (\mathbf{U}_k, \mathbf{U}_k) + (\mathbf{\Delta U}, \mathbf{\Delta V}) \tag{A1}$$

In a similar way, the graph regularization of (6) can be rewritten as follows:

$$\begin{aligned} L(\mathbf{\Delta U},\ \mathbf{\Delta V}\ ) = tr\left(\left(\mathbf{U}^T + \mathbf{\Delta U}^T\right) \mathbf{H}_u \left(\mathbf{U} + \Delta\mathbf{U}\right)\right) \\ tr\left(\left(\mathbf{V}^T\ \mathbf{\Delta V}^T\right) \mathbf{H}_v \left(\mathbf{V} + \mathbf{\Delta V}\right)\right) \end{aligned} \tag{A2}$$

With (7) and (8), our task is to minimize the following:

$$\begin{aligned} &\min_{\Delta\mathbf{U},\Delta\mathbf{V}} H_k\left(\Delta\mathbf{U}, \Delta\mathbf{V}\right) = \\ &\min_{\Delta\mathbf{U},\Delta\mathbf{V}} \parallel \mathbf{W} \odot \left(\mathbf{M} - \left(\mathbf{U}_k + \Delta\mathbf{U}\right) \left(\mathbf{V}_k^T + \Delta\mathbf{V}_T\right)\right) \parallel_1 \\ &\quad + \frac{\lambda}{2} \left(\parallel \mathbf{U} + \Delta\mathbf{U} \parallel_F^2 + \parallel \mathbf{V} + \Delta\mathbf{V} \parallel_F^2\right) \\ &\quad + \frac{\alpha}{2} L(\Delta\mathbf{U},\ \Delta\mathbf{V}\ ) \end{aligned} \tag{A3}$$

Now our task is to find a small increment $(\Delta\mathbf{U},\ \Delta\mathbf{V})$ such that the objective function keeps decreasing. Inspired by [26], we try to relax (9) to a convex surrogate.

By using the triangular inequality of norms, we arrive at the following inequality:

$$\begin{aligned} &H_k\left(\mathbf{\Delta U}, \mathbf{\Delta V}\right) \\ &\quad \leq \parallel \mathbf{W} \odot \left(\mathbf{M} - \mathbf{U}_k\mathbf{V}_k^T - \mathbf{\Delta U}\mathbf{V}_k^T - \mathbf{U}_k\mathbf{\Delta V}^T\right) \parallel_1 \\ &\quad + \frac{\lambda}{2} \left(\parallel \mathbf{U} + \Delta\mathbf{U} \parallel_F^2 + \parallel \mathbf{V} + \Delta\mathbf{V} \parallel_F^2\right) \\ &\quad + \frac{\alpha}{2} L\left(\Delta\mathbf{U}, \Delta\mathbf{V}\right) + \parallel \mathbf{W} \odot \Delta\mathbf{U}\Delta\mathbf{V}^T \parallel_1 . \end{aligned} \tag{A4}$$

Besides, we can introduce the following relaxation:

$$\begin{aligned} &\parallel \mathbf{W} \odot \left(\Delta\mathbf{U}\Delta\mathbf{V}^T\right) \parallel_1 \\ &\leq \frac{1}{2} \parallel \mathbf{\Lambda}_u\Delta\mathbf{U} \parallel_F^2 + \frac{1}{2} \parallel \mathbf{\Lambda}_v\Delta\mathbf{V} \parallel_F^2 . \end{aligned} \tag{A5}$$

For simplicity, we define $J_k\left(\Delta\mathbf{U}, \Delta\mathbf{V}\right)$ as follows:

$$\begin{aligned} &J_k\left(\Delta\mathbf{U}, \Delta\mathbf{V}\right) \\ &\quad = \parallel \mathbf{W} \odot \left(\mathbf{M} - \mathbf{U}_k\mathbf{V}_k^T - \mathbf{\Delta U}\mathbf{V}_k^T - \mathbf{U}_k\mathbf{\Delta V}^T\right) \parallel_1 \\ &\quad + \frac{\lambda}{2} \left(\parallel \mathbf{U} + \mathbf{\Delta U} \parallel_F^2 + \parallel \mathbf{V} + \mathbf{\Delta V} \parallel_F^2\right) \\ &\quad + \frac{\alpha}{2} L(\mathbf{\Delta U},\ \mathbf{\Delta V}). \end{aligned} \tag{A6}$$

Then we have the relaxed function of $H_k\left(\Delta\mathbf{U},\Delta\mathbf{V}\right)$. Our optimization problem can be recast as:

$$\begin{aligned} F_k\left(\Delta\mathbf{U},\Delta\mathbf{V}\right) &= J_k\left(\Delta\mathbf{U},\Delta\mathbf{V}\right) \\ &+\frac{1}{2}\parallel \mathbf{\Lambda}_u\Delta\mathbf{U}\parallel_F^2+\frac{1}{2}\parallel \mathbf{\Lambda}_v\Delta\mathbf{V}\parallel_F^2. \end{aligned} \tag{A7}$$

Thus, our optimization problem (9) can be further rewritten as:

$$\begin{aligned} &\min_{\mathbf{E},\Delta\mathbf{U},\Delta\mathbf{V}} \parallel \mathbf{W}\odot\mathbf{E}\parallel_1 \\ &+(\frac{\lambda}{2}\parallel \mathbf{U}+\Delta\mathbf{U}\parallel_F^2+\frac{1}{2}\parallel \mathbf{\Lambda}_u\Delta\mathbf{U}\parallel_F^2) \\ &+(\frac{\lambda}{2}\parallel \mathbf{V}+\Delta\mathbf{V}\parallel_F^2+\frac{1}{2}\parallel \mathbf{\Lambda}_v\Delta\mathbf{V}\parallel_F^2) \\ &+\frac{\alpha}{2}L(\Delta\mathbf{U},\Delta\mathbf{V}) \\ &s.t. \quad \mathbf{M}-\mathbf{U}_k\mathbf{V}_k^T=\mathbf{E}+\Delta\mathbf{U}\mathbf{V}_k^T+\mathbf{U}_k\Delta\mathbf{V}^T, \end{aligned} \tag{A8}$$

where $\mathbf{\Lambda}_u$, $\mathbf{\Lambda}_v$ are diagonal matrices.

We optimize the objective by the Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty (LADMPSAP) [45], as follows.

Appendix A.1. Updating E

Fixing other variables, updating **E** is equivalent to the following problem:

$$\min_{\mathbf{E}} \parallel \mathbf{W}\odot\mathbf{E}\parallel_1+\parallel \mathbf{E}-\ \mathbf{E}^i+\hat{\mathbf{Y}}^i/\delta_e^{(i)}\ \parallel_F^2. \tag{A9}$$

where

$$\hat{\mathbf{Y}}^i=\mathbf{Y}^i+\beta^i(\mathbf{E}^i+\Delta\mathbf{U}^i\mathbf{V}_k^T+\mathbf{U}_k\Delta\mathbf{V}^{iT}-\ \mathbf{M}+\mathbf{U}_k\mathbf{V}_k^T), \tag{A10}$$

and $\delta_e^{(i)}\ =\ \eta_e\beta^i,\eta_e\ =\ 3L_e+\varepsilon$, where 3 is the number of variables which have to be updated in parallel, such as **E**, $\Delta\mathbf{U}^i$, and $\Delta\mathbf{V}^i$. Specially, L_e is the squared spectral norm of the linear mapping on **E**, which is equal to 1, and ε is a small positive scalar. Then we update **E** by:

$$\mathbf{E}^{i+1}\ =\ \mathbf{W}\odot\mathbf{S}_{\sigma_e{(i)}}(\mathbf{E}^i-\ \hat{\mathbf{Y}}^i/\delta_e^{(i)})+\bar{\mathbf{w}}\odot(\mathbf{E}^i\ -\ \hat{\mathbf{Y}}^i/\delta_e^{(i)}), \tag{A11}$$

where **S** is the shrinkage operator [50]:

$$\mathbf{S}_\varepsilon\left(x\right)=max(|x|-\varepsilon,0)sgn(x), \tag{A12}$$

where $\bar{\mathbf{w}}$ is the complement of **W**.

Appendix A.2. Updating $\mathbf{\Delta U}$

Updating $\mathbf{\Delta U}$ is to solve the following problem:

$$\begin{aligned} &\min_{\mathbf{\Delta U}}\frac{\lambda}{2}\parallel \mathbf{U}_k+\mathbf{\Delta U}\parallel_F^2 \\ &+\frac{\alpha}{2}tr((\mathbf{U}_k^T+\mathbf{\Delta U}^T)\mathbf{H}_u(\mathbf{U}_k+\mathbf{\Delta U}))+\frac{1}{2}\parallel \mathbf{\Lambda}_u\mathbf{\Delta U}\parallel_F^2) \\ &+\frac{\delta_u^{(i)}}{2}\parallel \mathbf{\Delta U}-\mathbf{\Delta U}^i+\hat{\mathbf{Y}}^i\mathbf{V}_k/\delta_u^{(i)}\parallel_F^2 \end{aligned} \tag{A13}$$

where $\delta_u^{(i)} \; = \; \eta_u \beta^i$ and $\eta_u \; = \; 3 \parallel \mathbf{V}_k \parallel_2^2 + \varepsilon$. Since all terms in (A13) is convex, (A13) is a convex problem and its closed solution can be obtained by:

$$\begin{aligned}\Delta \mathbf{U}^{i+1} \; = & \\ & (\lambda \mathbf{I}_m + \alpha \mathbf{H}_u + \mathbf{\Lambda}_u^T \mathbf{\Lambda}_u + \delta_u^{(i)} \mathbf{I}_m)^{-1} \\ & (-\lambda \mathbf{U}_k - \alpha \mathbf{U}_k \mathbf{H}_u + \delta_u^{(i)} \mathbf{\Delta U}^i - \delta_u^{(i)} \hat{\mathbf{Y}}^i \mathbf{V}_k / \delta_u^{(i)}),\end{aligned} \tag{A14}$$

where m can be found in the paper [26].

Appendix A.3. Updating $\mathbf{\Delta V}$

Similar to $\mathbf{\Delta U}$, updating $\mathbf{\Delta V}$ can be achieved by:

$$\begin{aligned}\Delta \mathbf{V}^{i+1} = & \\ & (\lambda \mathbf{I}_m + \alpha \mathbf{H}_v + \mathbf{\Lambda}_v^T \mathbf{\Lambda}_v + \delta_v^{(i)} \mathbf{I}_m)^{-1} \\ & (-\lambda \mathbf{V}_k - \alpha \mathbf{V}_k \mathbf{H}_v + \; \delta_v^{(i)} \mathbf{\Delta V}^i - \delta_v^{(i)} \hat{\mathbf{Y}}^i \mathbf{U}_k / \delta_v^{(i)}).\end{aligned} \tag{A15}$$

Appendix A.4. Updating $\mathbf{Y}$ *and* β

We update $\mathbf{Y}$ and β as follows:

$$\begin{aligned}\mathbf{Y}^{i+1} = \mathbf{Y}^i + \beta^i (\mathbf{E}^{i+1} + \Delta \mathbf{U}^{i+1} \mathbf{V}_k^T \\ + \mathbf{U}_k \Delta \mathbf{V}^{(i+1)T} \mathbf{U}_k \mathbf{V}_k^T - \mathbf{M}),\end{aligned} \tag{A16}$$

$$\beta^{i+1} = min(\beta^{max}, \rho \beta^i), \tag{A17}$$

where ρ is defined by:

$$\rho \; = \begin{cases} \rho_0, & if \; \mathbf{Q} < \varepsilon_1 \\ 1, & otherwise, \end{cases} \tag{A18}$$

and

$$\begin{aligned}\mathbf{Q} = \beta^i \; max(\sqrt{\eta_e} ||\mathbf{E}^{i+1} \; - \; \mathbf{E}^i||_F, \\ \sqrt{\eta_u} ||\Delta \mathbf{U}^{i+1} \; - \; \Delta \mathbf{U}^i||_F, \\ \sqrt{\eta_v} ||\Delta \mathbf{V}^{i+1} \; - \; \Delta \mathbf{V}^i||_F) / ||\mathbf{M} - \mathbf{U}_k \mathbf{V}_k^T||_F.\end{aligned} \tag{A19}$$

In addition, the stopping criterion of iteration can be derived from KKT condition [45]:

$$\begin{aligned}\beta^i \max(\sqrt{\eta_e} ||\mathbf{E}^{i+1} \; - \mathbf{E}^i||_F, \\ \sqrt{\eta_u} ||\Delta \mathbf{U}^{i+1} \; - \; \Delta \mathbf{U}^i||_F, \\ \sqrt{\eta_v} ||\Delta \mathbf{V}^{i+1} \; - \; \Delta \mathbf{V}^i||_F) / ||\mathbf{M} - \mathbf{U}_k \mathbf{V}_k^T||_F \\ < \; \varepsilon_1,\end{aligned} \tag{A20}$$

$$\begin{aligned}||\mathbf{E}^{i+1} - \Delta \mathbf{U}^{i+1} \mathbf{V}_k^T - \mathbf{U}_k \Delta \mathbf{V}^{(i+1)T} \mathbf{U}_k \mathbf{V}_k^T||_F \\ / ||\mathbf{M} - \mathbf{U}_k \mathbf{V}_k^T||_F < \varepsilon_2.\end{aligned} \tag{A21}$$

Finally, Algorithm A1 is here rewritten in details as follows:

Algorithm A1 Graph Regularized Robust Matrix Factorization (GRMF) by Majorization Minimization

```
Input: $\mathbf{M} \in \mathbb{R}^{n\times m}$, $\alpha$, and $\lambda$
Output: $\mathbf{U}$ and $\mathbf{V}$
Method:
  Initialize $\mathbf{U}_0$ and $\mathbf{V}_0$ with using SVD on $\mathbf{M}$; $\mathbf{E}^0 = \mathbf{M} - \mathbf{U}_0\mathbf{V}_0^T$, and $\Delta\mathbf{U}^0 = \Delta\mathbf{V}^0 = \mathbf{Y}^0 = 0$. Besides,
  $\rho_0 = 1.5$. and $\varepsilon = \varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 1e - 5$.
  While not converged when we arrived $[\mathbf{U}_k, \mathbf{V}_k]$, do
      Let $t = 1$ and $\beta^0 =\propto (m+n)\varepsilon_1$;
      While (A20) and (A21) are not satisfied do
          Update $\mathbf{E}^t$ by (A11);
          Update $\Delta\mathbf{U}^t$ and $\Delta\mathbf{V}^t$ via (A14) and (A15);
          Update $\mathbf{Y}^t$ by (A16);
          Update $\beta^t$ by (A17);
          $t$=$t$+1;
      End while
      Update $\mathbf{U}$ and $\mathbf{V}$ in parallel:
          $\mathbf{U}_{k+1} = \mathbf{U}_k + \Delta\mathbf{U}_t$;
          $\mathbf{V}_{k+1} = \mathbf{V}_k + \Delta\mathbf{V}_t$;
      Check the convergence coditions, if
          $\mathbf{V}_{k+1} - \mathbf{V}_k < \varepsilon_2$ and $\mathbf{U}_{k+1} - \mathbf{U}_k < \varepsilon_3$;
  End while.
```

References

1. Shannon, G.; Kim, T. *Research Trends in Mathematics and Statistics*; AkiNik Publications: Delhi, India, 2019.
2. Iqbal, Z.; Qadir, J.; Mian, A.N.; Kamiran, F. Machine learning based student grade prediction: A case study. *arXiv* **2017**, arXiv:1708.08744.
3. Dietz-Uhler, B.; Hurn, J.E. Using learning analytics to predict (and improve) student success: A faculty perspective. *J. Interact. Online Learn.* **2013**, *12*, 17–26.
4. Zhang, Y.; Dai, H.; Yun, Y.; Shang, X. Student Knowledge Diagnosis on Response Data via the Model of Sparse Factor Learning. In Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019), Montreal, QC, Canada, 2–5 July 2019.
5. Moreno-Marcos, P.M.; Alario-Hoyos, C.; Muñoz-Merino, P.J.; Kloos, C.D. Prediction in MOOCs: A review and future research directions. *IEEE Trans. Learn. Technol.* **2018**, *12*, 384–401. [CrossRef]
6. Mayilvaganan, M.; Kalpanadevi, D. Comparison of classification techniques for predicting the performance of students academic environment. In Proceedings of the 2014 IEEE International Conference on Communication and Network Technologies, Sivakasi, India, 18–19 December 2014; pp. 113–118.
7. Elbadrawy, A.; Studham, R.S.; Karypis, G. Collaborative multi-regression models for predicting students' performance in course activities. In Proceedings of the Fifth International Conference on Learning Analytics and Knowledge, Poughkeepsie, NY, USA, 16–20 March 2015; pp. 103–107.
8. Zhang, Y.P.; Liu, S.H. Ensemble classification based on feature drifting in data streams. *Comput. Eng. Sci.* **2014**, *36*, 977–985.
9. Cortez, P.; Silva, A.M.G. *Using Data Mining to Predict Secondary School Student Performance*; EUROSIS-ETI, ETI Bvba: Ostend, Belgium, 2008.
10. Helal, S.; Li, J.; Liu, L.; Ebrahimie, E.; Dawson, S.; Murray, D.J.; Long, Q. Predicting academic performance by considering student heterogeneity. *Knowl.-Based Syst.* **2018**, *161*, 134–146. [CrossRef]
11. Yu, H.F.; Lo, H.Y.; Hsieh, H.P.; Lou, J.K.; McKenzie, T.G.; Chou, J.W.; Chung, P.H.; Ho, C.H.; Chang, C.F.; Wei, Y.H.; et al. Feature engineering and classifier ensemble for KDD cup 2010. In Proceedings of the KDD Cup, Washington, DC, USA, 25 July 2010.

12. Zhang, Y.; Xiang, M.; Yang, B. Linear dimensionality reduction based on Hybrid structure preserving projections. *Neurocomputing* **2016**, *173*, 518–529. [CrossRef]
13. Wang, T.; Mitrovic, A. Using neural networks to predict student's performance. In Proceedings of the International Conference on Computers in Education, Auckland, New Zealand, 3–6 December 2002; pp. 969–973.
14. Yang, T.Y.; Brinton, C.G.; Joe-Wong, C.; Chiang, M. Behavior-based grade prediction for MOOCs via time series neural networks. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 716–728. [CrossRef]
15. Su, Y.; Liu, Q.; Liu, Q.; Huang, Z.; Yin, Y.; Chen, E.; Ding, C.; Wei, S.; Hu, G. Exercise-enhanced sequential modeling for student performance prediction. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
16. Polyzou, A.; Karypis, G. Grade prediction with course and student specific models. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Auckland, New Zealand, 19–22 April 2016; pp. 89–101.
17. Thai-Nghe, N.; Drumond, L.; Horváth, T.; Krohn-Grimberghe, A.; Nanopoulos, A.; Schmidt-Thieme, L. Factorization techniques for predicting student performance. In *Educational Recommender Systems and Technologies: Practices and Challenges*; IGI Global: Pennsylvania, PA, USA, 2012; pp. 129–153.
18. Zhang, Y.P.; Chai, Y.M.; Wang, L.M. Method of concept drifting detection based on martingale in data stream. *J. Chin. Comput. Syst.* **2013**, *34*, 1787–1792.
19. Thai-Nghe, N.; Schmidt-Thieme, L. Multi-relational factorization models for student modeling in intelligent tutoring systems. In Proceedings of the 2015 Seventh International Conference on Knowledge and Systems Engineering (KSE), HoChiMinh City, Vietnam, 8–10 October 2015; pp. 61–66.
20. Koren, Y.; Bell, R.M.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *IEEE Comput.* **2009**, *42*, 30–37. [CrossRef]
21. Thainghe, N.; Drumond, L.; Krohngrimberghe, A.; Schmidtthieme, L. Recommender system for predicting student performance. *Conf. Recomm. Syst.* **2010**, *1*, 2811–2819.
22. Hwang, C.S.; Su, Y.C. Unified clustering locality preserving matrix factorization for student performance prediction. *IAENG Int. J. Comput. Sci.* **2015**, *42*, 245–253.
23. Lee, D.D.; Seung, H.S. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*; Massachusetts Institute of Technology Press: Cambridge, MA, USA, 2001; pp. 556–562.
24. Thai-Nghe, N.; Drumond, L.; Horváth, T.; Nanopoulos, A.; Schmidt-Thieme, L. Matrix and Tensor Factorization for Predicting Student Performance. In Proceedings of the 3rd International Conference on Computer Supported Education (CSEDU 2011), Noordwijkerhout, The Netherlands, 6–9 May 2011; Nguyen, T.-N., Lucas, D., Tomá, H., Alexandros, N., Lars, S.-T., Eds.; pp. 69–78.
25. Lorenzen, S.; Pham, N.; Alstrup, S. On predicting student performance using low-rank matrix factorization techniques. In Proceedings of the European Conference on e-Learning, Porto, Portugal, 26–27 October 2017; pp. 326–334.
26. Lin, Z.; Xu, C.; Zha, H. Robust matrix factorization by majorization minimization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 208–220. [CrossRef]
27. Zhang, Y.; Liu, S.; Shang, X.; Xiang, M. Low-rank graph regularized sparse coding. In Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Nanjing, China, 28–31 August 2018; pp. 177–190.
28. Zhang, Y.; Xiang, M.; Yang, B. Low-rank preserving embedding. *Pattern Recognit.* **2017**, *70*, 112–125. [CrossRef]
29. Rao, N.; Yu, H.F.; Ravikumar, P.K.; Dhillon, I.S. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems*; Massachusetts Institute of Technology Press: Cambridge, MA, USA, 2015; pp. 2107–2115.
30. Xu, J.; Moon, K.H.; Van Der Schaar, M. A machine learning approach for tracking and predicting student performance in degree programs. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 742–753. [CrossRef]
31. Egalite, A.J. How family background influences student achievement: Can schools narrow the gap? *Educ. Next* **2016**, *16*, 70–79.
32. Liu, S.; Shang, X. Hierarchical similarity network fusion for discovering cancer subtypes. In Proceedings of the International Symposium on Bioinformatics Research and Applications, Beijing, China, 8–11 June 2018; pp. 125–136.

33. Koprinska, I.; Stretton, J.; Yacef, K. Predicting student performance from multiple data sources. In Proceedings of the International Conference on Artificial Intelligence in Education, Madrid, Spain, 22–26 June 2015; pp. 678–681.
34. Saa, A.A. Educational data mining & students' performance prediction. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 212–220.
35. Févotte, C. Majorization-minimization algorithm for smooth Itakura-Saito nonnegative matrix factorization. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; pp. 1980–1983.
36. Wei, E.; Ozdaglar, A. Distributed alternating direction method of multipliers. In Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), Maui, HI, USA, 10–13 December 2012; pp. 5445–5450.
37. Hwang, C.R. Simulated annealing: Theory and applications. *Acta Appl. Math.* **1988**, *12*, 108–111.
38. Kalofolias, V.; Bresson, X.; Bronstein, M.; Vandergheynst, P. Matrix completion on graphs. *arXiv* **2014**, arXiv:1408.1717.
39. Brecko, B.N. How family background influences student achievement. In Proceedings of the IRC-2004 TIMSS, Nicosia, Cyprus, 11–13 May 2004; Volume 1, pp. 191–205.
40. Wenglinsky, H. Teacher classroom practices and student performance: How schools can make a difference. *ETS Res. Rep. Ser.* **2001**, *2001*, i37. [CrossRef]
41. Cai, D.; He, X.; Han, J.; Huang, T.S. Graph regularized nonnegative matrix factorization for data representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 1548–1560.
42. Goyal, P.; Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowl.-Based Syst.* **2018**, *151*, 78–94. [CrossRef]
43. Zhang, Y.; Xiang, M.; Yang, B. Graph regularized nonnegative sparse coding using incoherent dictionary for approximate nearest neighbor search. *Pattern Recognit.* **2017**, *70*, 75–88. [CrossRef]
44. Zhang, Y.; Xiang, M.; Yang, B. Hierarchical sparse coding from a Bayesian perspective. *Neurocomputing* **2018**, *272*, 279–293. [CrossRef]
45. Liu, R.; Lin, Z.; Su, Z. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. In Proceedings of the Asian Conference on Machine Learning, Canberra, ACT, Australia, 13–15 November 2013; pp. 116–132.
46. Rendle, S. Factorization machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 995–1000.
47. Jiang, B.; Lu, Z.; Li, N.; Wu, J.; Jiang, Z. Retweet prediction using social-aware probabilistic matrix factorization. In Proceedings of the International Conference on Computational Science, Wuxi, China, 11–13 June 2018; pp. 316–327.
48. Wong, N.C.; Lam, C.; Patterson, L.; Shayegan, B. Use of machine learning to predict early biochemical recurrence after robot-assisted prostatectomy. *BJU Int.* **2019**, *123*, 51–57. [CrossRef]
49. Sweeney, M.; Lester, J.; Rangwala, H. Next-term student grade prediction. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 970–975.
50. Lin, Z.; Chen, M.; Ma, Y. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv* **2010**, arXiv:1009.5055.

applied sciences

MDPI

Article

Use of Deep Multi-Target Prediction to Identify Learning Styles

Everton Gomede [1,*], Rodolfo Miranda de Barros [2] and Leonardo de Souza Mendes [1]

[1] Electrical Engineering and Computer College, State University of Campinas, Av. Albert Einstein, 400, Cidade Universitária Zeferino Vaz, Distrito Barão Geraldo, Campinas 13083-852 SP, Brazil; lmendes@decom.fee.unicamp.br

[2] Computer Science Department, State University of Londrina, Rod. Celso Garcia Cid, Km 380, s/n, Campus Universitário, Londrina 86057-970 PR, Brazil; rodolfo@uel.br

* Correspondence: evertongomede@gmail.com; Tel.: +55-(44)-99966-0874

Received: 28 January 2020; Accepted: 26 February 2020; Published: 4 March 2020

Featured Application: Our results can be applied to identifying of students' learning style providing adaptation to e-learning systems.

Abstract: It is possible to classify students according to the manner they recognize, process, and store information. This classification should be considered when developing adaptive e-learning systems. It also creates a comprehension of the different styles students demonstrate while in the process of learning, which can help adaptive e-learning systems offer advice and instructions to students, teachers, administrators, and parents in order to optimize students' learning processes. Moreover, e-learning systems using computational and statistical algorithms to analyze students' learning may offer the opportunity to complement traditional learning evaluation methods with new ones based on analytical intelligence. In this work, we propose a method based on deep multi-target prediction algorithm using Felder–Silverman learning styles model to improve students' learning evaluation using feature selection, learning styles models, and multiple target classification. As a result, we present a set of features and a model based on an artificial neural network to investigate the possibility of improving the accuracy of automatic learning styles identification. The obtained results show that learning styles allow adaptive e-learning systems to improve the learning processes of students.

Keywords: deep multi-target prediction; Felder–Silverman learning style; adaptive e-learning systems; artificial neural network; deep learning

1. Introduction

According to Willingham [1], people are normally curious but are not naturally acceptable masterminds; unless cognitive conditions are adequate, humans abstain from reasoning. This behavior is attributed to three properties. To begin with, reasoning is used with moderation; human's visual system is proficient to instantly take in a complex scene, although it is not inclined to instantly solve a puzzle. Additionally, reasoning is tiresome because it requires focus and concentration. Finally, because we ordinarily make mistakes, reasoning is uncertain. In spite of these aspects, humans like to think. Solving problems produces pleasure because there is an overlap between the brain's areas and chemicals that are important in learning and those related to the brain's natural reward system [1]. Thus, adjusting a student's cognitive style might help to improve the student's reasoning capacity.

Moreover, according to Felder and Silverman [2], learning styles (a part of cognitive styles) describe students' preferences on how some subject is presented, how to work with that subject matter, and how to internalize (acquire, process, and store) information [2]. According to Willingham [1], students may have diverse preferences on how to learn. Thus, knowing a student's learning style can help in finding the most suitable way to improve the learning process. There are some studies which show that learning styles allow adaptive e-learning systems to improve the learning processes of students [3–7].

We characterize learning as the procedure through which information is encoded and stored in the long-term memory. For instance, customizing content to the learning styles of students is seen as useful to learning in different ways, for example, improving fulfillment, increasing learning results, and decreasing learning time [3]. Several research works in learning systems proposed that students' learning improved when the instructor's teaching style matched the learning style of the students [5].

According to Bernard et al. [5], there are several methods to classify learning styles. One of the most known is the Felder–Silverman learning styles model (FSLSM). This model proposes four dimensions to classify learning styles, with each dimension classified in an interval ranging from 0 to 11. The first dimension of this model is active/reflective, which determines if someone prefers first experimenting with some subject and then reasoning about it (active) or reasoning first and then experimenting on the subject (reflective). The second dimension is sensing/intuitive, which determines if someone prefers touching things to learn (sensing) or observing things to induce information (intuitive). The third dimension, visual/verbal, determines if someone prefers to see charts, tables, and figures (visual) instead of reading or listening to texts (verbal), or the contrary. Finally, the sequential/global dimension determines if someone prefers to get information in a successive manner, learning step-by-step (sequential), or to get an outline of the information first and go into detail afterwards, without a predefined sequence (global).

According to Willingham [1], the prediction of any learning styles theory is that a particular teaching method might be good for some students but not so good for others. Therefore, it is possible to take advantage of different types of learning. It is important to understand the difference between learning ability and learning style. Learning ability is the capacity for or success in certain types of thought (math, for example). In contrast to ability, learning style is the tendency to think in a particular way, for example, thinking sequentially or holistically, which is independent of context [1]. There is an enormous contrast between the popularity of learning styles in education and evidence of their usefulness. According to Pashler [4], the reasonable utility of using learning styles to improve student learning still needs to be demonstrated. However, in their study, Kolb and Kolb [6] examined recent research advancements in experimental learning in higher education and analyzed how it can improve students' learning. In their studies they concluded that learning styles can be based on research and clinical observation of learning styles' score patterns and applied throughout the educational environment through an institutional development program.

As indicated by Willingham [1], psychologists have made a few approaches to test this learning proposition, leading to some hypotheses. First, learning style is considered as stable within an individual. In other words, if a student has a particular learning style, that style ought to be a stable part of that student's cognitive makeup. Second, learning style ought to be consequential; therefore, using a specific learning style should have implications on the outcomes of the student's learning. Thus, learning styles theory must have the following three features: (1) a specific learning style should be a stable characteristic of a person; (2) individuals with different styles should think and learn differently; and (3) individuals with different styles do not, on average, differ in their ability. Traditionally, learning styles are mainly measured through surveys and questionnaires, where students are asked to self-evaluate their own behaviors. However, this approach presents some problems. First, external interference can disturb the results during its application. Second, the outcomes are influenced by the quality of the survey or questionnaire. Finally, different students may interpret questions in different ways [5].

According to Bernard et al. [4], the characterization of learning styles is a problem that deals with many descriptors and many outputs. The descriptors may arise from many sources, such as logs, questionnaires, and databases. Moreover, the descriptors are usually associated with learning objects, such as forums, contents, outlines, quizzes, self-assignments, examples, and other types of resources. The outputs are used to permit the comprehension of learning style resulting from a combination of descriptors, which may indicate whether a student can be classified as active/reflective, sensing/intuitive, visual/verbal, or sequential/global, based on his/her approach to recognize, process, and store information. This problem is relevant because it is the first step to understand the cognitive condition to improve learning using e-learning systems [5].

In this context, our research aims to investigate the use of computational intelligence (CI) algorithms to analyze and improve the accuracy of autonomic approaches to identify learning style. Our hypothesis is that if the learning style can be correctly identified using CI then the student's learning preference may also be predicted. Thus, we conducted this research to identify features that may represent a student's learning style based on massive information (big data) collected in a massive open online course (MOOC) environment and use these features to classify these learning styles. We also investigate whether a theory of learning style might be more suited to classification than others. Finally, we investigate algorithms to overcome limitations found in contemporary works.

This paper is organized as follows: This first section presents basic considerations and justifications for this work and defines its main objectives. Section 2 presents an overall review of the key topics treated here and the main definitions upon which this work is based. Section 3 presents the main concepts behind learning styles classification and describes the proposed model. Section 3 also presents the data structures used to characterize the subjects, along with their materials and methods. Section 4 presents the results obtained from the data analysis and specifies recommendations to stakeholders. In the last section, conclusions and future developments are presented.

2. Related Work and Concepts

According to Truong [7] and Normadhi [3], researchers have been searching for mechanisms to automatically detect student's learning styles based on different models. The process of automatic learning style detection can be divided into three subproblems: (a) select a suitable learning model, (b) select the descriptors and targets to represent a student's online behavior (in a MOOC), and (c) select the algorithm (and hyperparameters) which fit to the multi-target prediction issue. This procedure is shown in Figure 1.

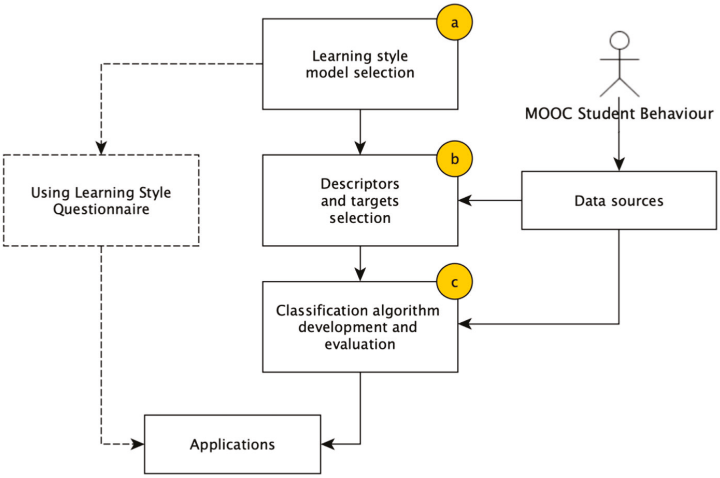

Learning style model selection
a
MOOC Student Behaviour
Using Learning Style Questionnaire
Descriptors and targets selection
b
Data sources
Classification algorithm development and evaluation
c
Applications

Figure 1. The process to build a model for automatic detection of learning style [5]. In this paper, we aim to investigate the steps a, b, and c. MOOC = massive open online course.

As shown in Figure 1, to classify students' learning styles, some researchers focused on the use of algorithms while others focused on the application of this model using traditional methods, such as questionnaires (dashed line). In this section, we compare papers that used these approaches.

2.1. Learning Styles Model Selection

According to Willingham [1], learning styles theory predicts that a particular teaching method may be good for one person, but not good for another. Therefore, in order to optimize a student's capacity to learn we need to exploit these different methods of learning. As previously said, it is imperative to comprehend the difference between learning ability and learning style. Learning ability is the capacity for, or success in, certain types of subjects (math, for example). In contrast to ability, learning style is a tendency to reason in a particular way, for example, sequentially or holistically. As already pointed out, there is a differentiation between the popularity of learning styles approaches within education and the lack of credible evidence for its utility. As indicated by Pashler [4], whether characterization of students' learning styles has any reasonable utility has yet to be determined. However, an investigation by Kolb and Kolb [6] examined ongoing advancements in the hypotheses and research on experiential learning and explored how it can help improve learning in higher education. In addition, Kolb and Kolb presumed that learning styles are based on both research and clinical observation of the patterns of learning styles' scores and can be applied throughout the educational environment by an institutional development program.

Various components of learning styles have been researched, both conceptually and empirically [3]. In addition, numerous hypotheses and multiple taxonomies attempting to describe how people reason and learn have been proposed, arranging individuals into distinct groups. Moreover, as indicated by Omar et al. [8], different learning style instruments to research and pedagogical purposes have been produced. From FSLSM theory, there are four dimensions that describe learning styles: processing, perception, input, and understanding. According to Truong [7] and Normadhi [3], many researchers developed automatic detection models based on the FSLSM, whose four dimensions are directly derived from its four objectives—processing, perception, input, and understanding. The processing dimension characterizes the active and reflective learners which are identified by their interest in performing physical or theoretical activities. Active students are the individuals who prefer to work in groups and perform numerous activities whereas reflective students prefer to work alone and perform some exercises. The sensitive and the intuitive learners are characterized by the perception dimension. Sensitive learners are those who are more attentive and careful and normally achieve their goals with few trials, presenting a high rate of exercise completion and reaching high performance in exams. On the other side, intuitive learners often become bored by details, show carelessness, and only achieve their goals with several trials presenting a low rate of exercise completion and reaching low performance in exams. The input measurement recognizes students by their inclination upon the visual or the verbal content and processes when studying and participating in group activities. Finally, the understanding dimension decides whether students incline towards the sequential or global methodology on understanding subjects of study. Sequential students prefer to move toward study and information in a sequential manner, similar to a road map, whereas global students prefer to get an overview and afterward dive into details, attempting to comprehend specific points and link that information with others [9].

To start the automatic learning style determination, the initial step is to choose a reasonable learning styles model. Nowadays, more than 70 models have been proposed, with some overlapping in their selection approaches. According to Truong [7], these models present some issues in terms of validity and reliability, with most of them presenting similar performances. From these, the Felder–Silverman learning styles model (FSLSM) is frequently used for automatic learning styles identification. Graf et al. [10] presented three reasons to select the FSLSM: (a) it uses four dimensions, allowing for more detailed classification; (b) it describes the preference to gather, process, and store information; and (c) it deals with each dimension as a tendency instead of an absolute type. These dimensions can be seen as a continuum with one learning inclination on the extreme left and the other on the extreme right, as per Saryar [11].

2.2. Descriptors and Target Selection

There are three primary sources of features that are recognized: log files, static information, and other personalization sources. The potential sources of data and the corresponding characteristics can be summarized as follows:

- Log files: this source collects users' behavior when they are interacting with MOOC. It is possible, in this case, to obtain such information as the number of visits and the time spent in several learning objects such as content, outlines, self-assessments, exercises, quizzes, forums, questions, navigation, and examples [9,10,12–16].
- History and background data of users: these include information that is either static or slow varying such as personal features (gender, age, etc.). They are rarely incorporated in programmed classification, although past research indicated that these variables assume a fundamental role in determining learning styles [7].
- Other personalization sources: these may incorporate background knowledge, intelligent capability, cognitive attributes (working memory capacity, learning skills, processing speed, and reasoning capacity), study objectives, language, and motivation level, which, in some cases, can be considered nearby with learning styles [7].

Regardless of the several predictors considered, none of the research addressed how different attributes contribute to predicting learning styles. The finding of such comparisons can assume an important role in improving the efficiency of different predictions.

2.3. Classification Algorithm Development and Evaluation

One of the most popular strategies used to classify and evaluate is the rule-based algorithm, in which researchers interpret different styles, according to the hypotheses, into different statistical rules. This method is used in Bayesian network and naïve Bayes rules. Moreover, other algorithms such as artificial neural network, ant colony optimization, particle swarm optimization, genetic algorithm, and decision tree can also be applied for classification. Among these algorithms, the one that accomplishes the optimal accuracy is artificial neural network (ANN) [5]. The common manner used to evaluate the models is splitting the dataset into training and test sub-datasets [17].

2.4. Related Works

In a review paper, Truong [7] presents a study summarizing several works in an overview of models used for learning styles classification. This paper analyzed 51 works, dividing learning style classification into three subproblems. According to the author, the models can be categorized into those that change over the time, those that change over situations, and those that do not change. In addition, the utilization of learning styles provides instructors with a tool to comprehend their students. Truong also shows that there is an association between learning styles and career choices. Based on this, suggestions and direction to support profession path planning can be developed. The author also divides the studies into those that only classify learning style and those that make predictions based on descriptors provided by user behavior. This last one is used for personalization and recommendation in e-learning systems.

In another survey paper, Normadhi et al. [3] stated that the techniques used to recognize personality characteristics can be divided into three categories: (a) questionnaires, (b) computer-based detection, and (c) both. Computer-based identification strategies are most often used to improve obtaining personality trait data in a student profile by analyzing implicit user input. These techniques are considered more accurate than the questionnaire techniques because they respond quickly to changes in the learner's personality characteristics. Computer-based recognition techniques can be categorized as machine learning, non-machine learning, and hybrid. Additionally, computer-based recognition techniques can be important for new students since information is initially insufficient to construct an appropriate student profile. In addition, the authors state that most researchers use personality traits in the cognition learning domain category (62.82%), in the adaptive learning environment, and in the model dimension, which are frequently used in the Felder–Silverman model (FSLSM). The authors also claim that the results of identification techniques have a positive and large influence on adaptive learning environments. For example, exploring observational assessment for adaptive e-learning environments is especially relevant. Research that conducts experiments to compare the effectiveness and efficiency of identification techniques is additionally highly encouraged. Finally, future examinations ought to explore and investigate the strength and weaknesses of personality traits that map into the learning object and materials selected [3].

Bernard et al. [5] investigated four computational intelligence algorithms (artificial neural network, ant colony optimization, genetic algorithm, and particle swarm optimization) to improve the accuracy of learning style detection. As a result, the authors achieved an average accuracy of 80% using artificial neural network. The authors also pointed out the drawbacks of using questionnaires, such as (a) it is assumed those learners are motivated to fill out the questionnaire; (b) they will fill it out fully (without influence); and (c) they understand how they prefer to learn. The authors used the FSLSM and relevant behavior descriptors from Graf et al. [10]. The authors also linked these descriptors with learning styles indicating that each descriptor is associated with a learning style. These descriptors are based on different types of learning objects including content, outlines, examples, exercises, self-assessment, quizzes, and forums. These descriptors consider the time which a student spends on a certain type of learning object (e.g., content_stay) and how often a student visits a certain type of learning object (e.g., content_visit). Moreover, questions were classified based on whether they are about concepts, if they require details or a general view of knowledge, if they include graphics, or if they use text only. These questions also deal with developing or interpreting solutions. Further, the authors presented metrics to evaluate the results. The performance of the proposed approaches was measured using four metrics: (a) SIM (similarity), (b) ACC (accuracy), (c) LACC (the lowest accuracy); and (d) % Match (percentage of students identified with a reasonable accuracy).

Another original paper, Sheeba and Krishnan [9] proposed a way to deal with classifying students' learning style based on their learning behavior. This approach is based on a decision tree classifier for the development of significant rules which are required for accurately distinguishing learning styles. This approach was experimented on 100 students for an online course created in the Moodle learning management system (LMS). In this experiment the authors accomplished the average accuracy of 87% in process, perception, and input dimension. The authors also presented two methods utilized for automatic recognition of learning styles: data-driven and literature-based approaches. The data-driven approach uses sample data to build a classifier that memorizes a learning style instrument. This approach predominantly uses artificial intelligence (AI) classification algorithm which takes the learner model as input and returns learners' learning style preferences as output. The literature-based approach utilizes simple rules to calculate learning styles from the quantity of matching hints. They used a dataset from web log files containing all the behaviors that the learner performed in Moodle LMS. These logs were automatically created when the students used the system. It records all the activities of forums, chats, exercises, assignments, quizzes, exam delivery, and frequency of accessing course materials [10].

Thus, our work aims to contribute to the papers analyzed here by proposing methods and procedures to overcome the current limitations. Here we first identify that the descriptors of previous studies are related with specific dimensions in the computational model. However, the psychological model, FSLSM [2], does not follow the same approach. For example, a student visits a course outline, activating the descriptor "outline_visit", which can be interpreted as a unique feature of the perception dimension (sensing/intuitive) [4,5]. Therefore, we investigate the interference of all the descriptors in the four dimensions using a multiple classification technique. Second, the strategy of labeling the dataset is vague. The logs provided for MOOC do not label learning style, only the behavior. The authors do not provide a clear method to label the dataset used in the training model [4,5]. Moreover, they do not present common problems related to datasets, such as imbalanced datasets [18]. Third, the context of dataset is not described in a comprehensible way. For example, the authors present the information of students' level (undergraduate students), however they do not indicate the average age, type of course, duration of course, frequency, and results (pass/fail) [4,5]. Moreover, with respect to computational intelligence techniques the authors do not provide an explicit strategy to overcome overfitting problems, a strategy to achieve optimal parameters (such as number of hidden layers in an artificial neural network), and a strategy to train and test the built models. Finally, there is a lack of performance metrics, such as f-score, recall, precision, sensitivity, and others [4,5]. This harms the possibility of comparing with related works (current and future) and does not present a different analysis from the test results.

3. Materials and Methods

The manner of integrating learning styles into an adaptive e-learning system may be divided into two essential areas: the build of a learning styles prediction model using online data (or the online learning styles classification model) and the application of this built model to an adaptive e-learning system. The development begins with choosing the learning styles model, for example, FSLSM. This is followed by determining the data sources and the learning styles attributes, and classification algorithm selections. After the evaluation, the suitable classification models and their outcomes are carried out for specific factors of the adaptive e-learning system.

The first step to build a model based on a computational intelligence algorithm is to collect and prepare a dataset. The students' behavior was collected from an LMS (learning management system) developed specifically for this experiment. The learning objects used were content, outlines, self-assessments, exercises, quizzes, forums, questions, navigation, and examples. The behavior was collected as described in Table 1. The 100 students graduated in Computer Science and enrolled in a post-graduate program in Computer Science and Project Management. The 26 descriptors were based on the Sheeba and Krishnan [9] and Bernad et al. [5] models. These descriptors were grouped into nine learning objects which are presented to the student in an LMS course. The dataset was composed of three types of measure: (a) "count", which represents the number of times a student visits a learning object; (b) "time", which represents the time the student spends in a learning object; and (c) "Boolean", which represents the students' results when responding to questions on a quiz. This record was collected for 15 days, and to summarize all the results obtained by the students, each descriptor was represented by the average of the students' logs. The questions on the self-assessment quizzes were categorized based on whether they are about facts or concepts, require knowledge about details or overview knowledge, include graphics, charts or text only, and address building or interpreting solutions. Table 1 shows the descriptors that were collected from the LMS. These descriptors also are considered as independent variables to build our model.

The resulting dataset does not provide a description of a learning style for each student. This information is necessary to train an algorithm based on supervised learning [5]. To overcome this problem, we used the Felder–Silverman questionnaire, (the original questionnaire which we used can be viewed at [19]) an adaptation to collect each student's learning style. This questionnaire classifies a student in FSLSM using four dimensions: (a) processing (active/reflective); (b) perception (sensing/intuitive); (c) input (visual/verbal); and (d) understanding (sequential/global). This classification is constructed defining a range for each dimension (for example, processing) from (−11:0) (active) to (0:11) (reflective), and so forth. The dataset's labels are shown in Table 2. These labels are also considered as dependent variables in our model.

Table 1. Descriptors of behavior collected from massive open online course (MOOC) (or independent variables).

Learning Object	Behavior	Measure [1]	Description
Content	content_visit	Count	Number of times that student visits a content
	content_stay	Time	Time spent (in seconds) at content
Outline	outline_stay	Time	Time spent (in seconds) at outline
	outline_visit	Count	Number of times that student visits an outline
Self-assessment	selfass_visit	Count	Number of times that student visits a self-assessment
	selfass_stay	Time	Time spent (in seconds) at self-assessment
	selfass_twice_wrong	Boolean	If student tests the self-assessment
Exercise	exercise_visit	Count	Number of times that student visits an exercise
	exercise_stay	Time	Time spent (in seconds) at exercise
Quiz	quiz_stay_results	Time	Time spent (in seconds) at quiz result
	quiz_revisions	Count	Number of times that student visits a quiz revision
Forum	forum_visit	Count	Number of times that student visits a forum
	forum_post	Count	Number of times that student posts a message in a forum
	forum_stay	Time	Time spent (in seconds) at forum
Questions	ques_detail	Time	Time spent (in seconds) at questions details
	ques_facts	Time	Time spent (in seconds) at questions from type facts
	ques_concepts	Time	Time spent (in seconds) at questions from type concept
	ques_develop	Time	Time spent (in seconds) at questions from type facts
	ques_graphics	Time	Time spent (in seconds) at questions from type graphics
	ques_text	Time	Time spent (in seconds) at questions from type text
	ques_overview	Time	Time spent (in seconds) at overview of questions
	ques_interpret	Time	Time spent (in seconds) at questions from type interpret
Navigation	navigation_skip	Count	Number of times that student skips navigation
	navigation_overview_visit	Count	Number of times that student visits navigation overview
	navigation_overview_stay	Time	Time spent (in seconds) at navigation
Example	example_stay	Time	Time spent (in seconds) at example

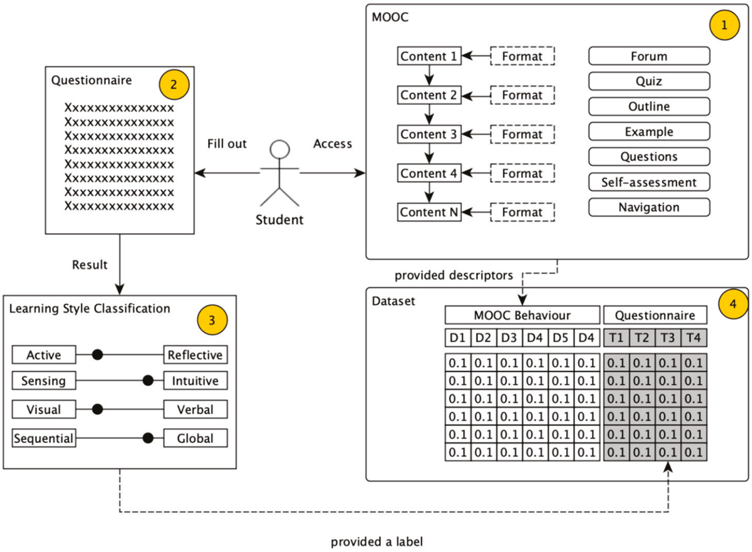
[1] The time and count are the average of everything accessed from a student for 15 days. Range of count >0 and range of time is from 0 to 120 s (this limit is because of session expiration time of MOOC).

Table 2. Labels of questionnaire (or dependent variables).

Dimension	Variable	Measure [1]	Description
Processing	active_reflective	Count	Value of student from −11 (active) to 11 (reflective)
Perception	sensing_intuitive	Count	Value of student from −11 (sensing) to 11 (intuitive)
Input	visual_verbal	Count	Value of student from −11 (visual) to 11 (verbal)
Understanding	sequential_global	Count	Value of student from −11 (sequential) to 11 (global)

[1] The count is the result of Felder–Silverman's questionnaire (0 values are not present).

The labels shown in Table 2 represent the students' learning behavior in the FSLSM. In these cases, the 0 values (or absence of preference) are not considered in labels because when a student fills out the questionnaire, he/she needs to choose the options which represent a dimension. The overall working flow for this process is shown in Figure 2, below.



MOOC Behaviour						Questionnaire			
D1	D2	D3	D4	D5	D4	T1	T2	T3	T4
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Figure 2. The process to build the dataset (independent and dependent variables). The questionnaire was used to label the dataset for each student's observation.

As shown in Figure 2, step 1 collects data from MOOC when a student interacts with a course. Then, in step 2 the system fills out the questionnaire for this student. In step 3, the results from the questionnaires based on the FSLSM are fed into a dataset. Finally, in step 4, the descriptors (independent variables) and labels (dependent variables) are combined with the raw dataset to produce an extended student classification dataset.

Since the dataset scale might be different for each student's measure (count, time, and Boolean), the next step to proceed with the dataset construction is to normalize the data to suitably compare information among students. Neural networks can be used to normalize data in order to improve their accuracy [5]. When analyzing two or more attributes it is often necessary to normalize the values of the attributes (for example, content_stay and content_visit), especially in those cases where the values are vastly different in scale. We use the range normalization [17] described in Equation (1):

$$\mathrm{x_i}' = \frac{\mathrm{x_i} - \mathrm{min_i}\{\mathrm{x_i}\}}{\mathrm{max_i}\{\mathrm{x_i}\} - \mathrm{min_i}\{\mathrm{x_i}\}} \tag{1}$$

After this transformation, the new attribute takes on values in the range (0, 1). Moreover, we converted the range of each dimension (processing, perception, input, and understanding) from (−11:0) and (0:11) to (0, 1). This transformation is required for two reasons: (a) the learning styles are a tendency [2,4], thus, to represent a student as active/reflective, we used a binary variable (e.g., active or reflective, instead of 11 times active or 11 times reflective) as a relaxation problem strategy, and (b) to improve the accuracy of the algorithm to classify four outputs. This operation is shown in Equation (2).

$$\begin{array}{c}\mathrm{IF\ (active\_reflective\ < 0)\ \{THEN\ active\_reflective = true\ ELSE\ false\}}\\ \mathrm{IF\ (sensing\_intuitive\ < 0)}\ \{\mathit{THEN\ sensing\_intuitive = true\ ELSE\ false}\}\\ \mathrm{IF\ (visual\_verbal\ < 0)\ \{THEN\ visual\_verbal = true\ ELSE\ false\}}\\ \mathrm{IF\ (sequential\_global\ < 0)}\ \{\mathit{THEN}\ \mathrm{sequential\_global = true\ ELSE\ false}\}\end{array} \tag{2}$$

In this case, each dimension receives TRUE to element at left and FALSE to element at right. For example, if a student's processing dimension is <0, then the student receives TRUE denoting that it is active. On the other hand, if a student's processing dimension is >0, then the student receives FALSE, denoting that it is reflective.

In addition, we investigate whether the dataset is imbalanced for each target. Imbalanced datasets mean the instances of one class is larger than the instances of another class (for example, more sequential rather than global in understanding dimension), where the majority and minority of class or classes are taken as negative and positive, respectively [11]. Figure 3 shows the distribution of each target.

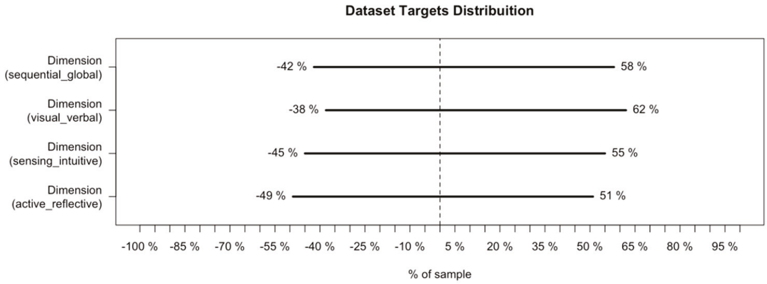


Figure 3. The dataset's target distribution of each dimension. The % of each student's preferences are represented, such a, sequential 42%, global 58%, visual 38%, verbal 62%, sensing 45%, intuitive 55%, active 49%, and reflective 51%.

As shown in Figure 3, this dataset does not have imbalanced data for any of the targets. For each dataset, the imbalance ratio (IR) is given by the division of the majority class by the minority class [18]. As a result, we obtained active_reflective (1.04), sensing_intuitive (1.22), visual_verbal (1.63), and sequential_global (1.38).

The algorithm chosen for multi-target prediction was artificial neural network (ANN) for five reasons: (a) there is evidence that this algorithm is better suited to solve learning style classification problems [5]; (b) since many authors use this algorithm, we can compare our results with other published ones [3]; (c) ANN works well with rather small datasets, which is important for this line of research considering that typical datasets are rather small [17]; (d) the problem can be translated to the network structure of an ANN; and (e) ANN allows multiple outputs analyzed at the same time. Moreover, the ANN architecture we used is feedforward multilayer perceptron, which means a neural network with one or more hidden layers [17,20].

The hidden layers act as feature detectors; as such, they play an important role in the operation of a multilayer perceptron. As the learning process advances throughout the multilayer perceptron, step by step the hidden neurons start to discover the features that describe the training data. They do so by performing nonlinear processing on the input data and transforming them into a new space, called the feature space. In this new space, the classes of interest in a pattern-classification task, for instance, may be more easily separated from each other than they could in the original input data space. Indeed, it is the creation of this feature space through supervised learning which distinguishes the multilayer perceptron from perceptron. Literature suggests that the number of hidden layers should be between log T (where T is the size of the training set) and 2× the number of inputs [17].

A popular approach for training multilayer perceptron is the back-propagation algorithm, which incorporates the least mean squares (LMS) algorithm as a special case. The training proceeds in two steps. In the first one, referred to as the forward phase, the synaptic weights of the network are updated and the input signal is propagated through the network, layer by layer, until the output. As a consequence, in this phase, adjustments are limited to the activation potentials and outputs of the neurons in the network. (b) In the second one, called the backward phase, an error signal is produced by evaluating the output of the network with an expected response. The resulting error signal is propagated through the network, again layer by layer, but this time the propagation is performed in the backward direction. In this second step, successive updates are made to the synaptic weights of the network. Calculation of the updates for the output layer is straightforward, but it is much more difficult for the hidden layers [17].

The back-propagation algorithm affords an approximation to the trajectory in weight space computed by the method of the stochastic gradient descent [17]. The smaller the value of the learning rate parameter α, the smaller the changes to the synaptic weights in the network. Consequently, it will be from the first iteration to the next in a smoother fashion during the trajectory in weight space. This improvement, however, is attained at the cost of a slower rate of learning. The learning rates used were between 0.01 and 0.1, in steps of 0.01, leading to the following values: (0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1) [17].

A training set is one of labeled data (for example, if a student is active or reflective) providing known information, which is used in supervised learning to build a classification or regression model. The training dataset is used to train the model (weights and biases in the case of artificial neural network) and then the model can see and learn from this data.

The model test is a critical but frequently underestimated part of model building and assessment. After preprocessing the data, they are needed to build a model with the potential to accurately predict further observations. If the built model completely fits the training data, it is probably not reliable after deployment in the real world. This problem is called overfit and needs to be avoided. A common manner on how to address the lack of an independent dataset for model evaluation is to reserve part of the learning data for this purpose. The basis for analyzing classifier performance is a confusion matrix (CF). This matrix describes how well a classifier can predict classes.

A typical cross validation technique is the k-fold cross validation. This method can be viewed as a recurring holdout method (holdout method divides the original dataset in two subsets, training and testing datasets). The whole data is divided into k equal subsets and each time a subset is assigned as a test set, the others are used for training the model. Thus, each observation gets a chance to be in the test and training sets; therefore, this method reduces the dependence of performance on test–training split and decreases the variance of performance metrics. Further, the extreme case of k-fold cross validation will occur when k is equal to the number of data points. It would mean that the predictive model is trained over all the data points except by one, which takes the role of a test set. This method of leaving one data point as a test set is known as leave-one-out cross validation (LOOCV) [17]. This technique is show in Figure 4.

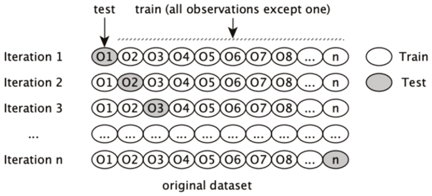


Figure 4. Leave-one-out cross validation technique (LOOCV). It is a special case of cross validation where the number of folds equals the number of instances in the dataset [17].

As shown in Figure 4, each iteration leaves one observation to test and the others to train. Therefore, the number of iterations is the number of observations in the dataset. The use of the leave-one-out procedure allows the model to be tested with all observations and prevents us from wasting these observations. This method was used to split the original dataset.

A classifier is evaluated based on performance metrics computed after the training process. In a binary classification problem, a matrix presents the number of instances predicted by each of the four possible outcomes: number of true positives (#TP), number of true negatives (#TN), number of false positives (#FP), and number of false negatives (#FN). Most classifier performance metrics are derived from the four values [21]. We used the following metrics in order to improve the accuracy of our model (Equations (3)–(14)) [22].

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{FN} + \text{TN}} \tag{4}$$

$$\text{Prevalence} = \frac{\text{TP} + \text{FP}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{5}$$

$$\text{PPV} = \frac{\text{sensitivity} * \text{prevalence}}{(\text{sensitivity} * \text{prevalence}) + ((1 - \text{specificity}) * (1 - \text{prevalence}))} \tag{6}$$

$$\text{NPV} = \frac{\text{specificity}*(1 - \text{prevalence})}{((1 - \text{sensitivity})*\text{prevalence}) + ((1 - \text{specificity}) * (1 - \text{prevalence}))} \tag{7}$$

$$\text{Detection Rate} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{8}$$

$$\text{Detection Prevalence} = \frac{\text{TP} + \text{FN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{9}$$

$$\text{Balanced Accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2} \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{12}$$

$$\text{F1} = \frac{\left(1 + \alpha^2\right)*\text{precicion} * \text{recall}}{(\alpha^2*\text{precision}) + \text{recall}} \tag{13}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{14}$$

For binary problems, the sensitivity, specificity, positive predictive value, and negative predictive value were calculated using the positive argument. The overall method is shown in Figure 5.

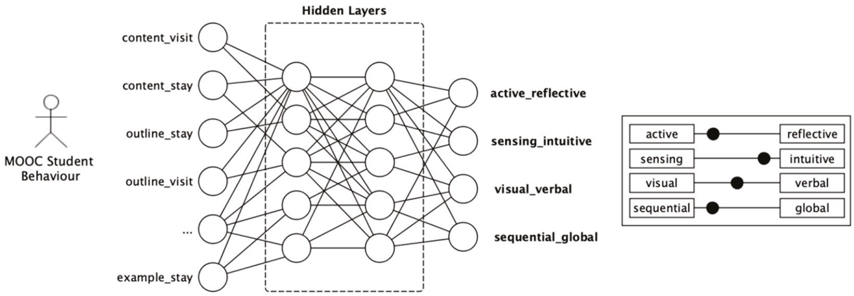


Figure 5. The overall method for learning styles classification based on count and time descriptors and targets. The behavior of the student is presented to the Multi Layer Perceptron (MLP) to train and get the weights which explain the labeled dataset.

As shown in Figure 5, the behavior of the 100 students is presented to the Multi Layer Perceptron (MLP) in order to train the neural network. The weight of each synapse (neuron connection) is obtained and the result is compared to the expected outcome (Equations (3)–(14)). When the accuracy (shown in

Table 3) is optimal (i.e., without improvement during the training step), the neural network training stops. The pseudocode that explains this method is presented below.

Pseudocode 1: The overall idea of method to learning style classification

```
$accuracy = 0$
$variation = max$
$while(variation > 0.01)\{$
    $present\ the\ subset\ of\ train\ and\ test\ to\ MLP$
    $weights = weights - delta\_weights * learning\_rate$
    $compare\ the\ predicted\ outcome\ to\ expected$
    $variation = the\ diference\ of\ the\ predicted\ and\ expected$
    $update\ accurary$
$\}$
```

Table 3. Metrics of each dimension.

Processing (Active_Reflective)	**Perception (Sensing_Intuitive)**
Accuracy: 0.85	Accuracy: 0.76
95% CI: (0.2306, 0.6847)	95% CI: (0.3605, 0.8088)
No Information Rate: 0.6	No Information Rate: 0.55
p-Value (Acc > NIR) [1]: 0.9435	*p*-Value (Acc > NIR): 0.4143
Kappa: −0.0784	Kappa: 0.1919
Mcnemar's Test *p*-Value: 0.5465	Mcnemar's Test *p*-Value: 1.0000
Sensitivity: 0.5000	Sensitivity: 0.5556
Specificity: 0.4167	Specificity: 0.6364
Pos Pred Value: 0.3636	Pos Pred Value: 0.5556
Neg Pred Value: 0.5556	Neg Pred Value: 0.6364
Prevalence: 0.4000	Prevalence: 0.4500
Detection Rate: 0.2000	Detection Rate: 0.2500
Detection Prevalence: 0.5500	Detection Prevalence: 0.4500
Balanced Accuracy: 0.4583	Balanced Accuracy: 0.5960
'Positive' Class: 1	'Positive' Class: 1
Input (Visual_Verbal)	**Understanding (Sequential_Global)**
Accuracy: 0.75	Accuracy: 0.80
95% CI: (0.4078, 0.8461)	95% CI: (0.1912, 0.6395)
No Information Rate: 0.65	No Information Rate: 0.65
p-Value [(Acc > NIR): 0.601	*p*-Value (Acc > NIR): 0.9940
Kappa: 0.2045	Kappa: −0.2371
Mcnemar's Test *p*-Value: 1.000	Mcnemar's Test *p*-Value: 0.7728
Sensitivity: 0.7692	Sensitivity: 0.4615
Specificity: 0.4286	Specificity: 0.2857
Pos Pred Value: 0.7143	Pos Pred Value: 0.5455
Neg Pred Value: 0.5000	Neg Pred Value: 0.2222
Prevalence: 0.6500	Prevalence: 0.6500
Detection Rate: 0.5000	Detection Rate: 0.3000
Detection Prevalence: 0.7000	Detection Prevalence: 0.5500
Balanced Accuracy: 0.5989	Balanced Accuracy: 0.3736
'Positive' Class: 1	'Positive' Class: 1

[1] NIR and ACC represent, respectively, No Information Rate (NIR) and Accuracy (ACC).

As shown in Pseudocode 1 all the subsets (train and test) are presented to the MLP to train and test. Accuracy is one of the metrics of Table 3 that is used to build the model, avoiding overfitting and underfitting [17].

4. Discussion

In this section, the results from the experiments are presented and discussed. We initially investigated the aspects of three types of variables: (a) count descriptors, (b) time descriptors, and (c) target descriptors where, this last one is of type count. There were no outliers found in the dataset. The median of the type count descriptors was around four accesses by the element (content_visit, outline_visit, etc.), as shown in Figure 4. The time descriptors define the time spent in each element (content_stay, outline_stay, etc.). The zero (0) value represents that the element had no access. The median time spent in type time was around 60 s and there was restriction that limited access at 120 s because of a time session limit. Finally, the target variables' median was around 0, which express the balanced learning styles dataset in each dimension (input, processing, perceive, and store), which means that the students are about evenly distributed between active and reflexive classes. These values are shown in Figure 6.

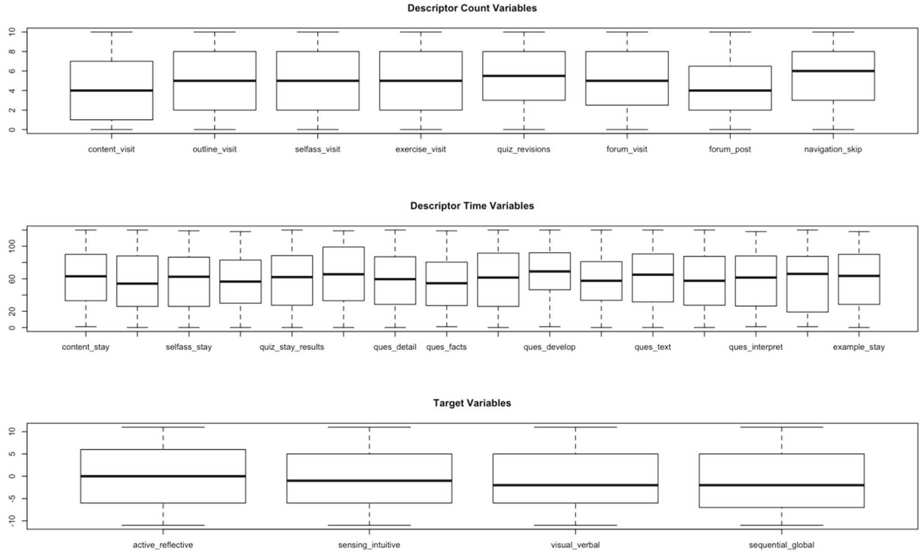


Figure 6. Box plot analysis of dataset (count and time descriptors and targets). There is a limitation to the time descriptor by 120 s and count by 10 times.

We also explored the frequency from each preference dimension before target transformation into binary variables (Equation (2)). As a result, we obtained the students' learning styles for each dimension. The dimensions active_reflective, sensing_intuitive, and sequential_global presented an approximately uniform distribution; however, the visual_verbal dimension presented a concentration close to −5, which represents a preference by the students to acquire visual information. Figure 7 shows this analysis.

We also investigated the possibility of using the dataset to identify students' preferences. If, for example, a determined set of attributes represents one of the four learning dimensions, these attributes may help in the dimensionality reduction and improve classifier precision [7,16]. The groups were investigated using the k-means algorithm to identify natural clusters in dataset. The k-means algorithm was used with k = {2, 3, 4, 5}. As a result, we obtained clusters with two and three groups with low overlap. By using a number of groups up to three, the resulting clusters overlapped. These results are shown in Figure 8.

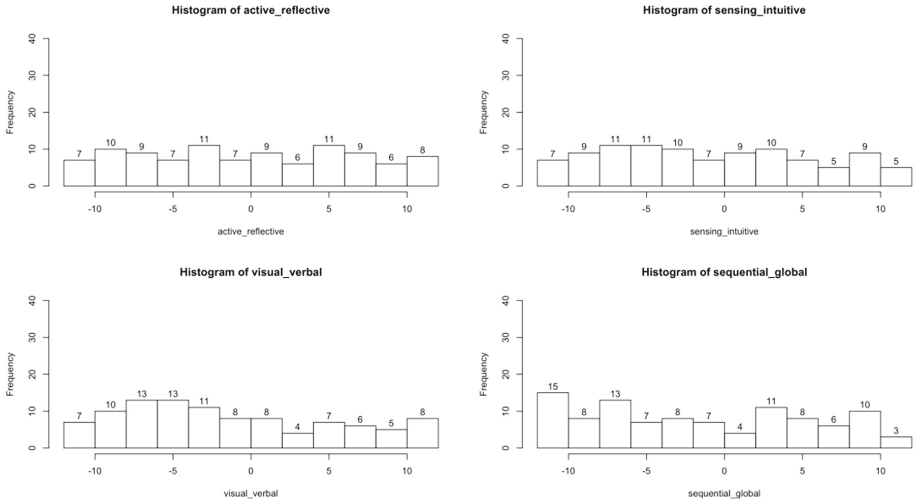


Figure 7. Learning style frequency distribution of each dimension.

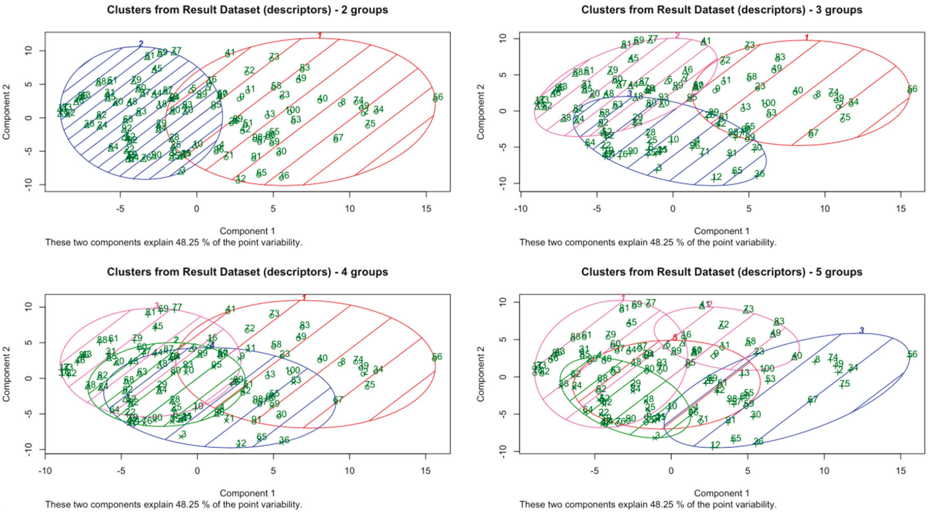


Figure 8. Natural clusters in dataset using k-means (with k = {2, 3, 4, 5}). No natural cluster was identified and, therefore, the classes are not linearly separated.

Additionally, another analytics technique, known as principal component analysis (PCA), was applied to investigate other relevant attributes or correlations and whether targets, of each dimension, might be explained by some descriptor (count and time). This is an important issue for dimensionality reduction in order to improve accuracy and reduce the cost of the model build [20]. These results are shown in Figure 9.

The dataset is balanced for the dimensions perceive, processing, and store, and presents some variation in the dimension input. In addition, there is not a predominant descriptor, making it possible to use all descriptors for the model construction.

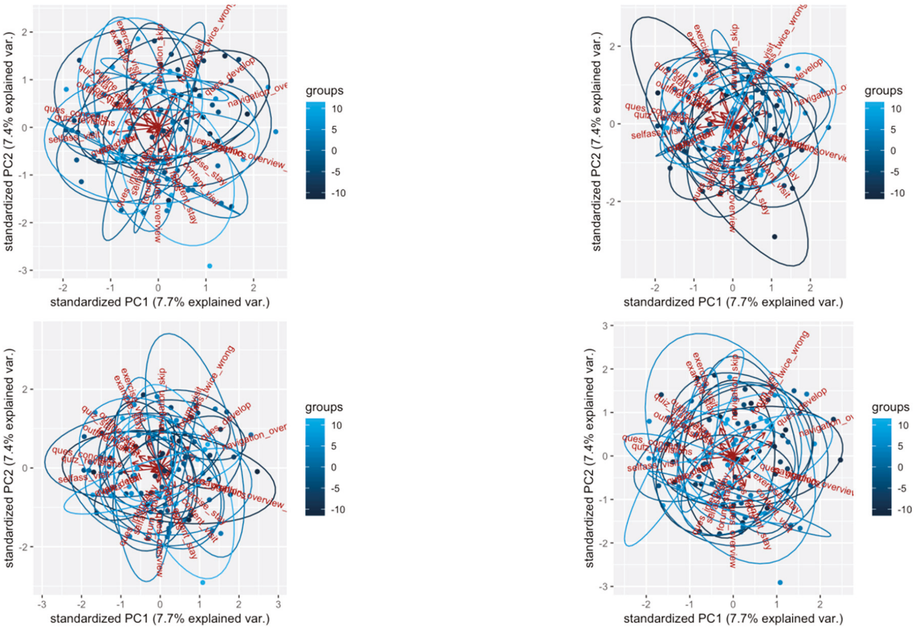


Figure 9. Principal component analysis (PCA) of each dimension, processing, perception, input, and understanding, respectively.

We may identify the onset of overfitting through the use of leave-one-out (special case of k-fold cross validation), for which the training data are split into an estimation subset and a validation subset. The estimation subset of examples is used to train the network in the usual way, except for a minor modification; the training session is stopped periodically (i.e., every so many epochs), and the network is tested on the validation subset after each period of training.

In our procedure, we varied the numbers of hidden layers for each model to determine a suitable number and provide the optimal result. The best model built presents two hidden layers, 26 neurons of input, and four neurons of output. The resulting model is shown in Figure 10.

This model evaluates all descriptors, simultaneously providing the students' classification in each learning dimension. This is a multi-target prediction algorithm [19]. Equations (4)–(14) were used to evaluate the model. We used the confusion matrix (CF) [17] to classify the predictions. The results of each dimension are shown in Table 3.

The best model built achieved 85%, 76%, 75%, and 80% accuracy in each target attribute, active_reflective, sensing_intuitive, visual_verbal, and sequential_global, respectively. These results are better than the results presented by Bernard et al. [5] (80%, 74%, 72%, and 82% in the same order) except for sequential_global, and simultaneously deal with all descriptors and targets instead of one at a time; we generated one model while Bernard et al. [5] generated four models to solve the problem. Moreover, we provided many performance metrics for each dimension to support further research to compare and improve their results (Table 3). In addition, we investigated the CF using area under the curve (AUC) and receiver operating characteristics (ROC). For each target the results were superior to Bernard et al. [5]. Figure 11 shows these results.

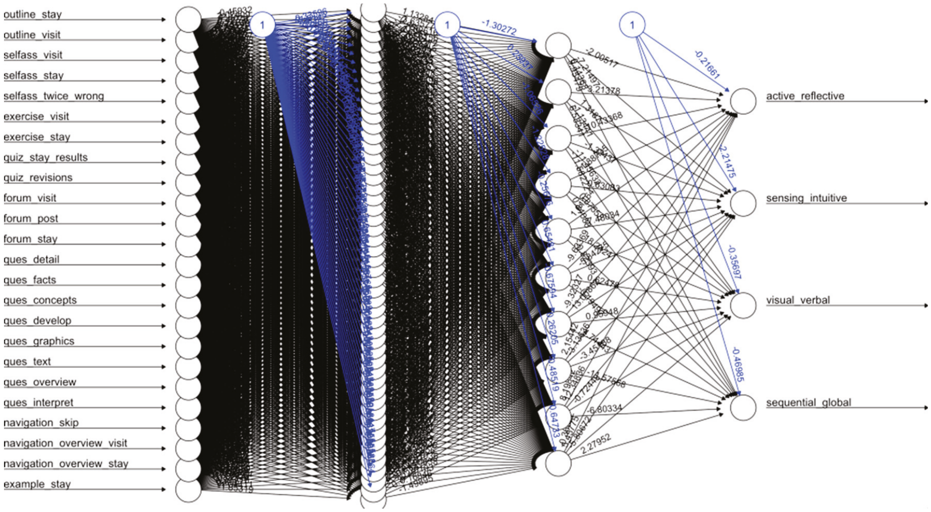


Figure 10. The built model with 2 hidden layers, 26 neurons of input, and 4 neurons of output.

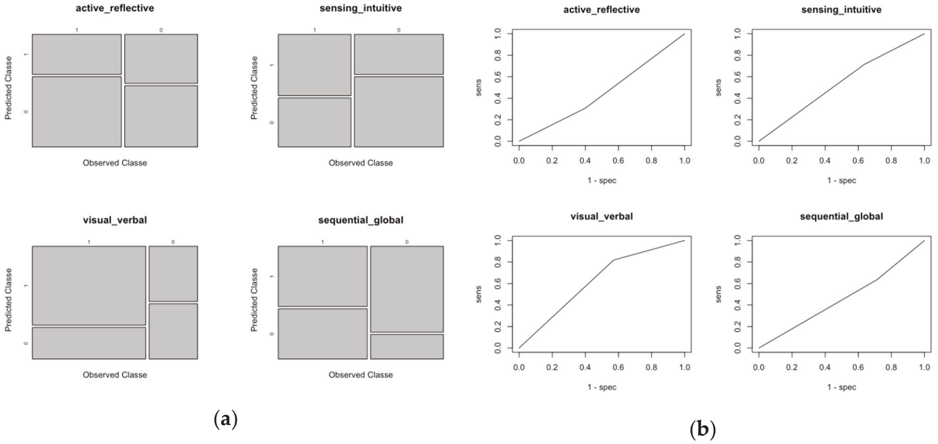


Figure 11. The receiver operating characteristics (ROC) analysis of each confusion matrix dimension that illustrates the diagnostic ability of the binary classifier system as its discrimination threshold is varied. (**a**) Represents the individual confusion matrix and (**b**) represents the individual ROC curve.

All metrics indicated that the model might be a method to automatically classify a student in a MOOC environment. The relation between descriptors improves the accuracy (as show in specificity and sensitivity from Table 3). Moreover, multi-target prediction (MTP) is a class of algorithms used with the simultaneous prediction of multiple target variables of diverse types, and the model using the Felder–Silverman procedure is by far, the most popular theory applied in adaptive e-learning system [5]. Meanwhile, from another point of view, the accuracy (and other performance metrics) of the outcomes using the proposed approach could be further improved by the use of a big dataset. Another limitation of the current research is that the results of the experiments were only tested on a platform with a specific subject (computer science and project management). The consistency of performance needs to be tested when it runs with different learning management systems and/or other online courses (for example, administration, economics, and so forth). Our future work will involve further exploration of the performance metrics and practical implications in different environments.

5. Conclusions and Further Development

This paper presents an automatic approach to identify learning styles of student behavior in a MOOC using a Computer Intelligent Algorithm (CIA) called deep artificial neural network (ANN). Assessment with the data of 100 students was performed, demonstrating the overall accuracy of the approach for each of the four Felder–Silverman learning styles model (FSLSM) learning style dimensions. It can be observed from the results obtained that this approach may be used to identify students' learning style based on their behavior in MOOC. This approach reduces the noise of questionnaires [3–5], allows classification when necessary to check if the style has changed over time [1], and allows data to be stored for future use.

Thus, by identifying students' learning styles, adaptive learning systems can use this information to provide more accurate personalization, leading to improved satisfaction and reduced learning time [3]. In addition, students can directly benefit from the more accurate identification of the learning styles, being able to leverage their strengths in relation to learning styles and understanding their weaknesses. In addition, teachers can use this learning style information to provide students with more accurate advice, which, as pointed out before, becomes more useful for students as learning style identification becomes more accurate as well. Moreover, students with similar learning styles may work together in the same classroom to improve their learning experience and help the teachers with their methods. Additionally, other stakeholders in the education ecosystem, such as parents, teachers, and administrators, can make use of such an approach to improve education in general [2,21].

Suggestions to further works may include the practical application of this approach through MOOC plug-ins. Different algorithms can be tested by comparing their results with works of the artificial neural network, since this work presents reference values based on the confusion matrix that can be replicated to other algorithms. Social issues can also be investigated to identify whether they influence learning styles. Concept drift (CD) should be investigated to identify if the target variables modify over time and compare the results to learning process questionnaires (LPQs). Finally, we may investigate how learning styles work in the propagation of information in networks based on complex networks [17]. This is an important topic, with great impact in real-world applications because it is a base to recommendation systems, and may be used to improve students' learning processes.

Author Contributions: This paper was developed with the active contribution of the doctorate candidate E.G., the orientation of L.d.S.M. and R.M.d.B. L.d.S.M. and R.M.d.B. are both advisors of the candidate E.G. and in addition to heading the research group, are mainly responsible for the propositions and hypotheses being proposed and tested. L.d.S.M. research areas are in Intelligent Cities and applied problems in Computational Intelligence. R.M.d.B. main research fields are in Governance, Strategic Planning, and applied Computational Intelligence. E.G. main research field is applied Computational Intelligence for Smart Cities. His main contribution to the paper was in processing data with computational intelligence techniques and mathematical models. All participants gave important contributions to the formulation of the hypotheses and the definition of the data analytics for the educational dataset. Based on The CRediT Roles, the contributions are E.G.: conceptualization, data curation, formal analysis, investigation, methodology, software, and writing—draft; R.M.d.B. and L.d.S.M.: project administration, resources, supervision, validation, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Willingham, D.T. *Why Don't Students Like School? A Cognitive Scientist Answers Questions about How the Mind Works and What It Means for the Classroom*; Jossey-Bass: San Francisco, CA, USA, 2010.
2. Felder, R.M.; Silverman, L.K. Learning and teaching styles in engineering education. *Eng. Educ.* **1988**, *78*, 674–681.

3. Afini Normadhi, N.B.; Shuib, L.; Md Nasir, H.N.; Bimba, A.; Idris, N.; Balakrishnan, V. Identification of personal traits in adaptive learning environment: Systematic literature review. *Comput. Educ.* **2019**, *130*, 168–190. [CrossRef]
4. Pashler, H.; Mcdaniel, M.; Rohrer, D.; Bjork, R. Concepts and Evidence. *Psychol. Sci. Public Interest* **2009**, *9*, 105–119. [CrossRef] [PubMed]
5. Bernard, J.; Chang, T.W.; Popescu, E.; Graf, S. Learning style Identifier: Improving the precision of learning style identification through computational intelligence algorithms. *Expert Syst. Appl.* **2017**, *75*, 94–108. [CrossRef]
6. Kolb, Y.A.; Kolb, A.D. Learning Styles and Learning Spaces: Enhancing Experiential Learning in Higher Education. *Acad. Manag. Learn. Educ.* **2016**, *4*, 193–212. [CrossRef]
7. Truong, H.M. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. *Comput. Hum. Behav.* **2016**, *55*, 1185–1193. [CrossRef]
8. Omar, N.; Mohamad, M.M.; Paimin, A.N. Dimension of Learning Styles and Students' Academic Achievement. *Procedia Soc. Behav. Sci.* **2015**, *204*, 172–182. [CrossRef]
9. Rasheed, F.; Wahid, A. Learning Style Recognition: A Neural Network Approach. In *First International Conference on Artificial Intelligence and Cognitive Computing, Advances in Intelligent Systems and Computing*; Springer: Singapore, 2019; Volume 815, pp. 301–312. [CrossRef]
10. Graf, S.; Liu, T.; Graf, S.; Liu, T. Supporting Teachers in Identifying Students' Learning Styles in Learning Management Systems: An Automatic Student Modeling Approach. *Int. Forum Educ. Technol. Soc.* **2016**, *12*, 2–14.
11. Saryar, S.; Kolekar, S.V.; Pai, R.M.; Pai, M.M.M. Mobile Learning Recommender System Based on Learning Styles. In *Soft Computing and Signal Processing, Advances in Intelligent Systems and Computing*; Springer: Singapore, 2019; Volume 898, pp. 299–312. [CrossRef]
12. Ahmad, N.; Tasir, Z.; Kasim, J.; Sahat, H. Automatic Detection of Learning Styles in Learning Management Systems by Using Literature-based Method. *Procedia Soc. Behav. Sci.* **2013**, *103*, 181–189. [CrossRef]
13. El-Bishouty, M.M.; Aldraiweesh, A.; Alturki, U.; Tortorella, R.; Yang, J.; Chang, T.W.; Graf, S. Use of Felder and Silverman learning style model for online course design. *Educ. Technol. Res. Dev.* **2019**, *67*, 161–177. [CrossRef]
14. Sheeba, T.; Krishnan, R. Automatic Detection of Students Learning Style in Learning Management System. In *Smart Technologies and Innovation for a Sustainable Future, Advances in Science*; Springer International Publishing: New York, NY, USA, 2019; pp. 45–53. [CrossRef]
15. Li, L.X.; Soraya, S.; Rahman, A. Students' learning style detection using tree augmented naive Bayes. *R. Soc. Open Sci.* **2018**, *5*, 17210. [CrossRef] [PubMed]
16. Franzoni, A.L.; Assar, S. Student learning styles adaptation method based on teaching strategies and electronic media. *Educ. Technol. Soc.* **2009**, *12*, 15–29.
17. Soudry, D.; di Castro, D.; Gal, A.; Kolodny, A.; Kvatinsky, S. Memristor-Based Multilayer Neural Networks with Online Gradient Descent Training. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2408–2421. [CrossRef] [PubMed]
18. Rout, N.; Mishra, D.; Mallick, M.K. Handling imbalanced data: A survey. *Adv. Intell. Syst. Comput.* **2018**, *628*, 431–443. [CrossRef]
19. Index of Learning Styles Questionnaire. Available online: https://www.webtools.ncsu.edu/learningstyles/ (accessed on 29 February 2020).
20. Liu, Y.-Y.; Slotine, J.-J.; Barabási, A.-L. Controllability of complex networks. *Nature* **2011**, *473*, 167–173. [CrossRef] [PubMed]
21. Gomede, E.; Gaffo, F.H.; Briganó, G.U.; de Barros, R.M.; Mendes, L.S. Application of Computational Intelligence to Improve Education in Smart Cities. *Sensors* **2018**, *18*, 267. [CrossRef] [PubMed]
22. Kuhn, M. Building predictive models in R using the caret package. *J. Stat. Softw.* **2008**, *28*, 1–26. [CrossRef]

applied sciences

MDPI

Article

Transfer Learning from Deep Neural Networks for Predicting Student Performance

Maria Tsiakmaki, Georgios Kostopoulos, Sotiris Kotsiantis * and Omiros Ragos

Department of Mathematics, University of Patras, 26504 Rio Patras, Greece; m.tsiakmaki@gmail.com (M.T.); kostg@sch.gr (G.K.); ragos@math.upatras.gr (O.R.)

* Correspondence: sotos@math.upatras.gr

Received: 14 February 2020; Accepted: 19 March 2020; Published: 21 March 2020

Abstract: Transferring knowledge from one domain to another has gained a lot of attention among scientists in recent years. Transfer learning is a machine learning approach aiming to exploit the knowledge retrieved from one problem for improving the predictive performance of a learning model for a different but related problem. This is particularly the case when there is a lack of data regarding a problem, but there is plenty of data about another related one. To this end, the present study intends to investigate the effectiveness of transfer learning from deep neural networks for the task of students' performance prediction in higher education. Since building predictive models in the Educational Data Mining field through transfer learning methods has been poorly studied so far, we consider this study as an important step in this direction. Therefore, a plethora of experiments were conducted based on data originating from five compulsory courses of two undergraduate programs. The experimental results demonstrate that the prognosis of students at risk of failure can be achieved with satisfactory accuracy in most cases, provided that datasets of students who have attended other related courses are available.

Keywords: transfer learning; deep learning; educational data mining; student performance prediction

1. Introduction

Transferring knowledge from one domain to another has gained a lot of attention among scientists in the past few years. Consider the task of predicting student performance (pass/fail) in higher education courses. According to the traditional supervised learning approach, a sufficient amount of training data, regarding a specific course C_A, is required for building an accurate predictive model which is subsequently used for making predictions on testing data derived from the same course. If the testing dataset is derived from a different course, C_B, sharing some common characteristics with course C_A (hereinafter referred to as related or similar courses), then transfer learning is the appropriate machine learning methodology for building accurate learning models in a more efficient manner, since it could contribute to the improvement of the predictive performance of the target domain model (course C_B) exploiting the knowledge of the source domain (course C_A) [1]. In a nutshell, a learning model is trained for a specific task using data derived from a source domain and, subsequently, it is reused for another similar task in the same domain or the same task in a different domain (target domain) [2,3]. More generally, when we lack information about a problem, we could train a learning model for a related problem, for which there is plenty of information, and apply it to the existing one.

Transfer learning is currently gaining popularity in deep learning [4]. Not long ago, it was claimed as the second "driver of machine learning commercial success", whereas supervised learning was the first one [5]. Pre-trained deep networks, usually trained on large datasets and thus requiring significant computation time and resources, are employed as the starting point for other machine learning problems due to their ability to be repurposed either for a new or for a similar task. Therefore, these networks could support complex problems in a more efficient way, since they can decrease the training time for building a new learning model and finally improve its generalization performance [6].

In recent years, several types of Learning Management Systems (LMSs) have been successfully adopted by universities and higher education institutions, recording a variety of student learning features and gathering huge amounts of educational data. Educational Data Mining (EDM) is a fast-growing scientific field offering the potential to analyze these data and harness valuable knowledge from them. To this end, a plethora of predictive algorithms have been effectively applied in educational contexts for solving a wide range of problems [7]. However, building predictive models in the EDM field through transfer learning methods has been poorly studied so far. Therefore, the main question in the present study is whether a predictive model trained on a past course would perform well on a new one. Boyer and Veeramachaneni observe that courses (a) might evolve over time in a dissimilar way, even if they are not much different in terms of context and structure, (b) are populated with different students and instructors, and (c) might have features that cannot be transferred (e.g., a feature defined on a specific learning resource which is not available on another course) [8]. In addition, the complexity of LMSs as well as the course design have a significant impact on the course progress during the semester [9]. Therefore, there may be problems where transfer learning might not reflect the anticipating results, showing some uncertainty about the predictive accuracy of the newly created learning model [10].

In this context, the present study aims to propose a transfer learning methodology for predicting student performance in higher education, a task that has been extensively studied in the field of EDM through traditional supervised methods. To this purpose, we exploit a set of five datasets corresponding to five undergraduate courses, each one lasting one semester, all supported by a Moodle platform. Initially, we form all the unique pairs of datasets (twenty pairs in total) matching the features of the paired courses one by one and generating new features if necessary. Next, a deep network model is trained by using the dataset of the first course and, subsequently, it is applied on the dataset of the second course for further training after a predefined number of epochs. Deep networks have been successfully applied in the EDM field for solving important educational problems, such as predicting student performance [11–14], dropout [15–17], or automatic feature extraction [18]. The main objective is to discover whether transfer learning accelerates training and improves the predictive performance utilizing the potential of deep neural networks in the EDM field. On this basis, we hope to provide a useful contribution for researchers.

The remainder of this paper is organized as follows. In the next section, we discuss the transfer learning approach, while in Section 3 we present an overview of some related studies in the EDM field. The research goal, together with an analysis of the datasets and description of the proposed transfer learning method, is set in Section 4. The experimental results are presented in Section 5, while Section 6 discusses the research findings. Finally, Section 7 summarizes the study, considering some thoughts for future work.

2. The Transfer Learning Approach

The traditional supervised learning methods exploit labeled data to obtain predictive models in the most efficient way. Let us consider the task of predicting whether a student is going to successfully pass or fail the examinations of an undergraduate course C_A. In this case, the training and the testing set are both derived from the same domain (course). The training set is used to build a learning model h by means of a classification algorithm (e.g., a deep network) and subsequently it is applied on the testing set for evaluating its predictive performance (Figure 1a). Some key requirements for achieving high performance models are the quality and sufficiency of the training data which are, unfortunately, not always easy to meet in real world problems. In addition, the direct implementation of model h for a different course C_B or a new task (e.g., predicting whether a student is going to drop out of the course) seems rather difficult. The existing model does not have the ability to generalize well to data coming from a different distribution, while, at the same time, it is not applicable, since the class labels of the two tasks are different.

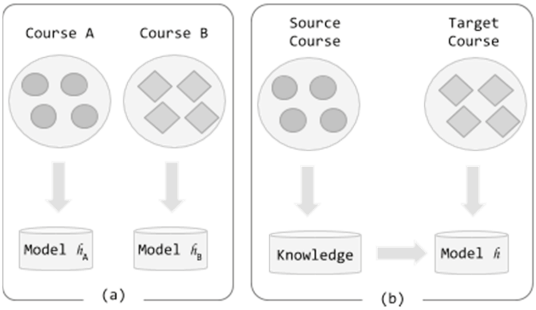


Figure 1. The traditional machine learning process (**a**), the transfer learning process (**b**).

Contrasting these methods, knowledge transfer or transfer learning intends to improve the performance of learning and provide efficient models in cases where data sources are limited or difficult and expensive to acquire [1,2], primarily due to their generalization ability to heterogeneous data (i.e., data from different domains, tasks and distributions [19]). Transfer learning might help us to train a predictive model h based on data derived from course C_A (source course) and apply it on data derived from a different but related course C_B (target course), which are not sufficient to train a model, for predicting the performance of a student. This indeed, is the aim of transfer learning: transfer the knowledge acquired from course C_A to course C_B and improve the predictive performance of model h (Figure 1b) instead of developing a totally new model, on the basis that both datasets should share some common attributes (i.e., common characteristics of students, such as their academic achievements or interactions within an LMS).

More formally, the transfer learning problem is defined as follows [1,20]:

A domain $\mathcal{D}$ is formed by a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$. A learning task $\mathcal{T}$ is formed by a label space Y and an objective predicted function $f(\cdot)$. The function f can be also written as $P(y|x)$, representing the conditional probability distribution of label y given a new instance x. $P(y|x)$ is learned from the training data $\{\mathcal{X}, \mathrm{Y}\}$. Given a source domain $\mathcal{D}_S = \{\mathcal{X}_S, P_S(\mathrm{X})\}$, its corresponding learning task $\mathcal{T}_S = \{\mathrm{Y}_S,\ f_S(\cdot)\}$, a target domain $\mathcal{D}_T = \{\mathcal{X}_T, P_S(\mathrm{X})\}$ and its corresponding learning task $\mathcal{T}_T = \{\Upsilon_T,\ f_T(\cdot)\}$, the purpose of transfer learning is to obtain an improved target predictive function $f_T(\cdot)$ by using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$. The fact that $\mathcal{D}_S \neq \mathcal{D}_T$ means that either $\mathcal{X}_S \neq \mathcal{X}_T$ or $P(X_S) \neq P(X_T)$, where $\mathrm{X}_{S_i} \in \mathcal{X}_S$ and $\mathrm{X}_{T_i} \in \mathcal{X}_T$. Similarly, the fact that $\mathcal{T}_S \neq \mathcal{T}_T$ means that either $\mathrm{Y}_S \neq \mathrm{Y}_T$ or $f_S(\cdot) \neq f_T(\cdot)$.

The inequalities contained in the definition form four different transfer learning settings:

- $\mathcal{X}_S \neq \mathcal{X}_T$: the feature space of the source and target domain are different. For example, the courses have different structure and context;
- $P(X_S) \neq P(X_T)$: the marginal probability distribution of the source and target domain are different. For example, the same courses offered by different departments or the same course offered in different years by the same department, thus consisting of different students;
- $\Upsilon_S \neq \Upsilon_T$: the label spaces of the source and target task are different (this setting usually occurs with setting four). For example, the source domain has two classes (e.g., {pass, fail}) and the target domain has six classes (e.g., {A, B, C, D, E, F});
- $f_S(\cdot) \neq f_T(\cdot)$: the conditional probability distributions of the source and target tasks are different. For example, the source and target courses are very unbalanced in relation to the defined classes.

Based on the above definition and conditions, three types of transfer learning settings are identified [1,8,18,21]: inductive transfer learning, transductive transfer learning and unsupervised transfer learning. In inductive transfer learning, the source domain is different but related to the target domain ($\mathcal{D}_S \neq \mathcal{D}_T$) regardless of the relationship between the tasks. In transductive transfer learning, both source and target task are the same ($\mathcal{T}_S = \mathcal{T}_T$), while the domains are different ($\mathcal{D}_S \neq \mathcal{D}_T$). Finally, in unsupervised transfer learning, the tasks are different ($\mathcal{T}_S \neq \mathcal{T}_T$), while both datasets do not contain labels. The latter type is intended for clustering and dimensionality reduction tasks.

3. Related Work

Predicting students' learning outcomes is considered one of the major tasks of the EDM field [22]. This is demonstrated by a great number of significant studies which put emphasis on the development and implementation of data mining methods and machine algorithms for resolving a plethora of predictive problems [23]. These problems are mainly intended to predict the future value of an attribute (e.g., students' grades, academic performance, dropout, etc.) based on a set of input attributes that describe a student. One typical problem is to detect whether a student is going to successfully pass or fail a course by the end of a semester based on his/her activity on the LMS, as in this study. The successful and accurate detection of students at risk of failure is of vital importance for educational institutions, since remedial measures and intervention strategies could be applied to support low performers and enhance their overall learning performance [24]. It is therefore necessary to build very accurate and robust learning models. Transfer learning could contribute to improving these models, since prior knowledge regarding a specific task could be useful to another similar task. Transfer learning is an approach which has still not been sufficiently examined in the field of EDM, as evidenced by the study of the current literature. To the best of our knowledge, there are few studies focusing on resolving prediction problems through transferring learning models from one domain to another, although this prospect is appealing. These studies indicate that building models based on a particular course and them applying to a new one (different but somehow related) is a rather complex task, which, unfortunately, does not always reflect the anticipating outcomes [10]. A list of some notable works regarding transfer learning in the EDM field are presented in the following paragraphs.

Ding et al. investigated the transferability of dropout prediction across Massive Online Open Courses (MOOCs) [9]. Therefore, they presented two variations of transfer learning based on autoencoders: (a) using the transductive principal component analysis, and (b) adding a correlation alignment loss term. The input data were click-stream log events of mixtures of similar and dissimilar courses. The proposed transfer learning methods proved to be quite effective for improving the dropout prediction, in terms of Area Under Curve (AUC) scores, compared to the baseline method. In a similar study, Vitiello et al. [25] examined how models trained on a MOOC system could be transferred to another. Therefore, they built a unified model allowing the early prediction of dropout students across two different systems. At first, the authors confirmed significant differences between the two systems, such as the number of active students and the structure of courses. After that, they defined a set of features based on the event logs of the two systems. Overall, three dropout prediction experiments were conducted: one for each separate system, one where each system applied a learning model built on the other system and one where the dataset contained data from both systems. The accuracy measure was above the baseline threshold (0.5) in most cases.

The method put forward by Hunt et al. [26] examined the effectiveness of TrAdaBoost, an extended AdaBoost version in the transfer learning framework, for predicting students' graduation rates in undergraduate programs. The dataset was based on a set of academic and demographic features (152 features in total) regarding 7637 students of different departments. Two separate experiments were conducted, each time using specific data for the training set. In the first experiment, the training set comprised all students apart from those studying engineering, while in the second one, the training set comprised all students that were suspended on academic warnings. The experimental results showed that the TrAdaBoost method recorded the smallest error in both cases. In the same context, Boyer and Veeramachaneni suggested two different approaches for predicting student dropout taking into account the selection method of the training data and how to make use of past courses information [8].

Therefore, several tests were performed using either all available information for a learner or a fixed subset of them. In addition, two different scenarios were formulated: inductive and transductive transfer learning. The experimental results indicated that the produced learning models did not always perform as intended. Very recently, Tri, Chau and Phung [27] proposed a transfer learning algorithm, named CombinedTL, for the identification of failure-prone students. Therefore, they combined a case-based reasoning framework and four instance-based transfer learning algorithms (MultiSource, TrAdaboost, TrAdaboost, and TransferBoost). The experimental results showed that the proposed method outperformed the single instance-based transfer learning algorithms. In addition, the authors compared the CombinedTL with typical case retrieval methods (k-NN and C4.5), experimenting with a varying number of target instances, finding that the performance of the proposed method was improved as the number of target instances was increased.

The notion of domain adaptation is highly associated with transfer learning. Zeng et al. [28] proposed a self-training algorithm (DiAd) which adjusts a classifier trained on the source domain to the target domain based on the most confident examples of the target domain and the most dissimilar examples of the source domain. Moreover, the classifier is adjusted to the new domain without using any labeled examples. Very recently, López-Zambrano et al. [29] investigated the portability of learning models based on Moodle log data regarding the courses of different universities. The authors explored whether the grouping of similar courses (i.e., similarity level of learning activities) influence the portability of the prediction models. The experimental results showed that models based on discretized datasets obtained better portability than those based on numerical ones.

4. Research Methodology

4.1. Research Goal

The main purpose of our study is to evaluate the effectiveness of transfer learning methods in the EDM field. More specifically, we investigate whether a deep learning model that has been trained using student data from one course can be repurposed for other related courses. Deep neural networks are represented by a number of connecting weights between the layers. During the training process, these weights are adjusted in order to minimize the error of the expected output. Therefore, the main notion behind the suggested transfer learning approach is to initialize a deep network using the pre-tuned weights from a similar course. Two main research questions guide our research:

(1) Can the weights of a deep learning model trained on a specific course be used as the starting point for a model of another related course?

(2) Will the pre-trained model reduce the training effort for the deep model of the second course?

4.2. Data Analysis

In the present study, we selected data regarding five compulsory courses of two undergraduate programs offered by the Aristotle University of Thessaloniki in Greece. More precisely, three courses (Physical Chemistry I (Spring 2018) and Analytical Chemistry Laboratory (Spring 2018, Spring 2019)) were offered by the department of Chemical Engineering, while two courses (Physics III (Spring 2018, Spring 2019)) were offered by the department of Physics. Table 1 provides detailed information regarding the gender and target class distribution of the five courses.

Table 1. Gender and target class distribution of the courses.

Course	Female		Male		Pass		Fail	
C1: Physical Chemistry I (Spring 2018)	122	43.3%	160	56.8%	134	47.5%	148	52.5%
C2: Physics III (Spring 2018)	90	50.0%	90	50.0%	74	41.1%	106	56.9%
C3: Analytical Chemistry Lab (Spring 2018)	57	48.2%	72	55.8%	105	81.4%	24	18.6%
C4: Physics III (Spring 2019)	80	50.6%	78	49.4%	68	43.0%	90	57.0%
C5: Analytical Chemistry Lab (Spring 2019)	61	52.1%	56	47.9%	100	85.5%	17	14.5%

Each course was supported by an online LMS, embedding a plethora of resources and activities. The course pages were organized into topic sections containing the learning material in the form of web pages, document files and/or URLs, while the default announcements forum was enabled for each course allowing students to post threads and communicate with colleagues and tutors. Each course required the submission of several assignments, which were evaluated on a grading scale from zero to 10. All sections were available to the students until the end of the semester, while the course final grade corresponded to the weighted average of the marks of all submitted assignments and the finishing exam. Note that successful completion of the course required a minimum grade of five.

For the purpose of our study, the collected datasets comprised six different types of learning resources: forums, pages, recourses, folders, URLs and assignments (Table 2). For example, course C_1 was associated with one forum, seven pages, 17 resources, two folders and eight assignments, three of which were compulsory. Regarding the forum module, we recorded the total number of views for each student. We also recorded the total number of times students accessed a page, a resource, a folder or a URL. Moreover, two counters were embedded in the course LMS, aggregating the number of student views (course total views) as well as the number of every type of recorded activity for a student (course total activity). Learning activities that were not accessed by students were not included in the experiments, while a student who did not access a learning activity was marked with a zero score. Finally, a custom Moodle plugin was developed, enabling the creation of the five datasets [30].

Table 2. Features extracted from students' low-level interaction logs.

Learning Resources	C_1	C_2	C_3	C_4	C_5	Description	Possible Values
Forum	1	1	1	1	1	Total number of times a student accessed the resource	0 or positive integer
Page	7	6	2	2	0		
Recourse	17	15	4	12	10		
Folder	2	0	17	12	0		
Url	0	0	1	1	0		
Assignments	8	9	8	8	9		
Submitted Assignments	3	9	8	8	9	Student Grades	[0, 10] decimal

It is worth noting that there were certain differences among the five courses (Tables 1 and 2). At first, they were offered by different departments (Physics and Chemical Engineering) and they had different format and content. Although courses C_2, C_4 and C_3, C_5 encompassed the same topic—that is, Physics and Chemistry, respectively—their content varied depending on the academic year of study. In addition, courses C_1, C_2, C_4 were theoretical (Physical Chemistry and Physics), while C_3, C_5 were laboratory courses (Analytical Chemistry Lab). Moreover, each course required the submission of a different number of assignments. Finally, it should be noted that different students attended these courses.

4.3. The Proposed Transfer Learning Approach

The present study intends to address the problem of transferring knowledge across different undergraduate courses. Hence, we employed a simple deep neural network architecture, comprised four layers: an input layer, two hidden dense layers and an output one. The input layer consists of input units corresponding to each one of the dataset input features (Table 3). The first hidden layer has 12 hidden units and the second one has eight. Both dense layers use the Relu activation function. Finally, the output layer consists of a single neuron employing the sigmoid activation function and the binary cross entropy loss function for predicting the output class (pass or fail).

The experimental procedure was divided into three distinct phases (Figure 2). In the first phase, we constructed all the unique pairs of courses that could be formed (ten pairs of courses in total). Each time, the related datasets were rebuilt to share a common set of features. For each pair of courses, we made use of the following notation:

$$\left\{C_i, C_j\right\}\ i, j \in \{1, 2, 3, 4, 5\},\ i \neq j. \tag{1}$$

Table 3. Features of the paired datasets.

	$\{C_1,C_2\}$	$\{C_1,C_3\}$	$\{C_1,C_4\}$	$\{C_1,C_5\}$	$\{C_2,C_3\}$	$\{C_2,C_4\}$	$\{C_2,C_5\}$	$\{C_3,C_4\}$	$\{C_3,C_5\}$	$\{C_4,C_5\}$
Forum views	1	1	1	1	1	1	1	1	1	1
Page views	7	7	8	7	0	0	2	0	2	2
Recourse views	17	17	17	17	11	11	11	10	12	12
Folder views	2	2	0	2	12	0	12	13	12	12
URL views	0	0	0	1	0	0	1	1	1	1
Assignments views	13	8	9	8	9	9	9	9	8	9
Submitted Assignments	8	8	9	8	8	9	8	9	8	9
Total views	1	1	1	1	1	1	1	1	1	1
Total activity	1	1	1	1	1	1	1	1	1	1
Gender	1	1	1	1	1	1	1	1	1	1
Total Number of Features	51	46	47	47	44	33	47	46	47	49

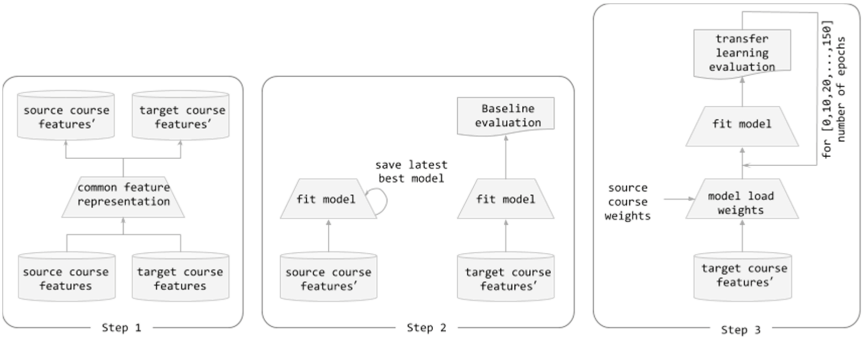


Figure 2. The three-step process of the proposed method.

In order to create a common set of features for each pair of courses, we matched features of the first course to related features of the second course one by one. Among the common features were the gender as well as the course total activity and course total views counters. Therefore, the first assignment of the first course was matched with the first assignment of the second course, the second assignment of the first course was matched with the second assignment of the second course and so forth, while the same procedure was followed for all the six types of resources. In cases where a matching feature was not found, a new feature was created, with zero values for each instance. For example, the C_1 course contained features related to seven page resources, whereas the C_2 course contained features related to six page resources (Table 2). Finally, the new $\{C_1,\ C_2\}$ pair of datasets contained seven features regarding the page resources, since a new empty feature was created and added in the C_2 course dataset, thus matching to the seventh feature of the C_1 course (Table 3).

The second phase refers to the training process of the two supporting deep networks. The first one was trained on the new source course C_i in order to extract its adjusted weights, while the second one was trained on the new target course C_j in order to calculate the baseline evaluation. In both cases, we calculated the accuracy metric, which corresponds to the percentage of correctly classified instances, while the models were trained for 150 epochs. In addition, the 10-fold cross validation resampling procedure was adopted for evaluating the overall performance of the deep network models.

The third phase was the most fundamental, since it implemented the transfer learning strategy. The deep model of the target course was fitted from scratch, but this time the network weights were initialized using the previously calculated weights from the source course (second phase). The pre-trained model was further tuned by running it each time for a certain number of epochs (hereinafter denoted as $C_{i,j}$): zero (i.e., the starting point), 10, 20, 30, 40, 50, 100 and 150. Algorithm 1 provides the pseudocode of the proposed transfer learning method. All the experiments were conducted using the Keras library in Python [31].

Algorithm 1: Transfer learning through paired course using deep neural networks.

```
Input: c1, c2, scores = [[] # c1 is the source course dataset, and c2 is the target
Output: scores # accuracy scores of the target course for every set of epochs
(c1′, c2′) ← commonRepresentation (c1, c2) # construct a common representation
model1 ← createAndCompileModel () # configure the deep learning process
weights ← fitModel (model1, c1′, epochs=150) # train the model, and save its weights
model2 ← createAndCompileModel(weights) # load the weights of pre-trained model1
for each  e in [0, 10, 20, 30, 40, 50, 100, 150] do
    model2′ ← fitModel (model2, c2′, epochs= e) # further tune the pre-trained model2
    score ← evaluate (model2′, c2′, folds=10) # evaluate model2′ using the accuracy metric
    add (score, scores) # save score for the output
end for each

# construct a common feature representation for the two courses
function commonRepresentation (dataset1, dataset2)
    dataset1′= [[], dataset2′= [[] # init empty datasets
    # match the features of dataset1 with the features of dataset2,
    # create new features when necessary
    for each t in [‘forum’, ‘page’, ’recourse’, ‘folder’, ‘url’, ‘assign views’, ‘assign’] do
        features1 ← getFeaturesOfType (dataset1, t) # get all features for this type
        features2 ← getFeaturesOfType (dataset2, t)
        size ← min (features1.size, features2.size)
        diff ← absoluteDifference (features1.size, features2.size)
        for i = 0 to size-1 do
            add (features1[i], dataset1’)
            add (features2[i], dataset2’)
        end for
        for j=0 to diff-1 do
            if f1 ← getFeatureAt(features1, features1.size + j) do # if f1 exists
                add (f1, dataset1’)
                add (createEmptyFeature(), dataset2’)
            else f2 ← getFeatureAt(features2, features2.size + j) # else f2 exists
                add (f2, dataset2’)
                add (createEmptyFeature(), dataset1’)
            end if
    end for
end for each
return dataset1′, dataset2′ # return the new datasets
```

5. Results

The averaged accuracy results (over the 10 folds) are presented in Table 4. For each pair, we conducted two experiments, using each course alternatively as the source course and the other one as the target course. Therefore, we evaluated 20 distinct combinations formed by the five courses. For each pair, we highlighted in bold the cases where the transfer model produced better results than the baseline. Overall, it is observed that the model $C_{i,j}$ benefits the predictions of the source course C_i, since the predictive performance of the transfer learning deep network is better than the baseline C_j.

Table 4. Averaged accuracy results.

	$\{C_1, C_2\}$		**$\{C_1, C_3\}$**		**$\{C_1, C_4\}$**		**$\{C_1, C_5\}$**		**$\{C_2, C_3\}$**	
	C_1	C_2	C_1	C_3	C_1	C_4	C_1	C_5	C_2	C_3
Baseline	0.7627	0.6094	0.7667	0.7047	0.7563	0.6333	0.7424	0.5591	0.6011	0.7144
Epochs	$C_{2,1}$	$C_{1,2}$	$C_{3,1}$	$C_{1,3}$	$C_{4,1}$	$C_{1,4}$	$C_{5,1}$	$C_{1,5}$	$C_{3,2}$	$C_{2,3}$
0	0.6106	**0.6172**	0.7524	**0.8227**	0.5675	**0.6371**	0.5889	**0.8644**	**0.6306**	0.6538
10	**0.7701**	**0.6414**	**0.7988**	**0.8382**	**0.8128**	**0.6387**	**0.8087**	**0.8561**	0.5975	**0.8234**
20	**0.7842**	**0.6299**	**0.7917**	**0.8537**	**0.8093**	0.5833	**0.8055**	**0.8553**	**0.6132**	**0.8382**
30	**0.7877**	**0.6132**	**0.8198**	**0.8537**	**0.8126**	0.6008	**0.8092**	**0.8644**	**0.6290**	**0.8394**
40	**0.7732**	**0.6234**	**0.8022**	**0.8608**	**0.8236**	0.6075	**0.8019**	**0.8553**	**0.6241**	**0.8453**
50	**0.7766**	0.6076	**0.8062**	**0.8465**	**0.8166**	0.6325	**0.7987**	**0.8386**	**0.6179**	**0.8537**
100	**0.7768**	0.5968	**0.7847**	**0.8465**	**0.8061**	**0.6762**	**0.7916**	**0.8114**	**0.6077**	**0.8394**
150	0.7552	**0.6185**	**0.7882**	**0.8620**	**0.7738**	0.6325	**0.7845**	**0.8371**	**0.6064**	**0.8472**
	$\{C_2, C_4\}$		**$\{C_2, C_5\}$**		**$\{C_3, C_4\}$**		**$\{C_3, C_5\}$**		**$\{C_4, C_5\}$**	
	C_2	C_4	C_2	C_5	C_3	C_4	C_3	C_5	C_4	C_5
Baseline	0.5650	0.6263	0.5498	0.6735	0.7096	0.6196	0.7715	0.7955	0.6811	0.6646
Epochs	$C_{4,2}$	$C_{2,4}$	$C_{5,2}$	$C_{2,5}$	$C_{4,3}$	$C_{3,4}$	$C_{5,3}$	$C_{3,5}$	$C_{5,4}$	$C_{4,5}$
0	**0.6207**	0.6008	0.4112	**0.8379**	0.4641	0.4988	**0.8156**	**0.8644**	0.5375	**0.6902**
10	**0.5941**	0.5817	**0.5916**	**0.8644**	**0.8310**	0.5892	**0.8156**	**0.8303**	0.5758	**0.7621**
20	**0.6154**	0.6133	**0.6031**	**0.8644**	**0.8394**	0.6067	**0.8239**	**0.8470**	0.5888	**0.8197**
30	**0.6120**	0.5946	**0.6188**	**0.8561**	**0.8465**	0.5563	**0.8322**	**0.8561**	0.6013	**0.8114**
40	**0.6074**	0.6192	**0.6130**	**0.8644**	**0.8406**	0.5883	**0.8310**	**0.8561**	0.6017	**0.8121**
50	**0.6330**	0.6258	**0.5938**	**0.8644**	**0.8549**	**0.6263**	**0.8251**	**0.8561**	0.6529	**0.8023**
100	**0.5885**	**0.6388**	**0.6272**	**0.8553**	**0.8322**	**0.6458**	**0.8329**	**0.8553**	0.6538	**0.7947**
150	**0.5947**	0.6133	**0.6105**	**0.7947**	**0.8251**	**0.6392**	**0.8299**	**0.8470**	0.6346	**0.8121**

A one-tailed, paired t-test (a=0.05) was conducted for verifying whether the improvement in the transfer model was statistically significant. Therefore, we compared the accuracy results obtained by the baseline deep network (using the target course dataset), with the results obtained by the transfer method, iteratively, for each number of epochs. Since the p-value is inferior or equal to 0.05, we conclude that the difference is significant in all cases except the starting point where the number of epochs equals zero (Table 5). Moreover, the *p*-value is gradually decreased as the number of epochs increase from every epoch from 10 to 100.

Table 5. The t-test results.

Epochs	***p*-Value**
0	0.2449
10	0.0051
20	0.0023
30	0.0022
40	0.0013
50	0.0003
100	0.0002
150	0.0012

The analysis of the experimental results, in question-and-answer format, underlines the efficiency of the proposed method for transferring knowledge from one course to a related one.

1. Can the weights of a deep learning model trained on a specific course be used as the starting point for a model of another related course?

At the starting point for each transfer learning model (i.e., zero epochs) we used the weights estimated by the previously trained deep network models (on 150 epochs) instead of starting with randomly initialized weights. For example, at the starting point of the $C_{4,2}$ transfer model, we used the weights estimated by the C_4 model.

Comparing the results of the pretrained weights without further tuning (i.e., zero epochs) to the baseline model, an improvement is noticed in half of the datasets (10 out of 20). The statistical results (t-test) confirm that the difference is not significant when the pre-trained model is not further tuned for the second dataset (target course C_j), since p-value=0.2449>a=0.05. However, the transfer model prevails in 16 out of 20 datasets when it is further tuned for only 10 epochs.

2. Will the pre-trained model reduce the training effort for the deep model of the second course?

Overall, the increase in the number of epochs improves the performance of the proposed transfer learning model. Moreover, the improvement is significant for every number of epochs, apart from the starting point, as statistically confirmed by the t-test results. It is worth noting that the transfer model prevails in 18 out of 20 datasets after 100 epochs, where the lowest p-value is 0.0002.

In addition, we can detect three cases of overfitting, since the accuracy ceases to improve after a certain number of iterations and begins to decrease. Particularly, this is observed in the cases where C_1 starts with C_2 weights, C_2 with C_1 weights and C_4 with C_1 weights. For instance, C_1 outperforms the baseline with an accuracy measure of 0.7768 after 100 epochs of retuning the preloaded weights of C_2. However, after 150 epochs the accuracy is decreased to 0.7552.

6. Discussion

An important finding to emerge in this study is that even a small amount of prior knowledge from a past course dataset could result in a fair measure of accuracy for predicting student performance in a related current course. This was verified by a plethora of experiments that have been carried out regarding twenty different pairs of five distinct one-semester courses, investigating the effectiveness of transfer learning in deep neural networks for the task of predicting at-risk students in higher education. In most cases, the transfer model obtained better accuracy than the baseline one. An improvement was noticed in half of the datasets (10 out of 20) using the pretrained weights from the source course (i.e., zero epochs). There was also a considerable accuracy improvement in most cases (16 out of 20) when the pre-trained model was further tuned for 10 to 40 epochs. Therefore, fine-tuning provides a substantial benefit over training with random initialization of the weights, thus leading to higher accuracy with fewer passes over the data. Overall, there was only one case where the transfer learning did not achieve better results ($C_{5,4}$). Hence, it is evident that it is not always feasible to transfer knowledge from one course to another one. In addition, it is worth noting that the type of course, laboratory or theoretical, does not seem to directly affect the predictive accuracy of the transfer learning model. This indicates that there is a slight uncertainty about the transferability level of a predictive model. The definition of what is a “transferable” model is where this ambiguity lies. A model trained on a set of courses is considered to be “transferable” if it achieves respectively fair results on a new, related course [10].

We believe this is yet another important attempt towards transfer knowledge in the educational field. Further, there are key issues to be considered such as measuring the degree of similarity between two courses (i.e., the number and form of learning activities), the type of attributes and the duration of the course. Finally, it is similarly important to build both simple and interpretable transferable models that could be easily applied by educators from one course to another [29]. Therefore, more studies are required on the current topic for establishing these results.

7. Conclusions

In the present study, an effort was made to propose a transfer learning method for the task of predicting student performance in undergraduate courses. The identification of failure-prone students could lead the academic staff developing learning strategies that aim to improve students’ academic performance [32]. Transfer learning enables us to train a deep network using the dataset of a past course (source course) and reuse it as the starting point for a dataset of a new related course (target course). Moreover, it is possible to further tune the repurposed model. Our findings proved that a fair performance was achieved in most cases, while the proposed method handily outperforms the baseline model.

Transfer learning offers many future research directions. Our results are encouraging and should be validated by larger samples of courses from different departments and programs. An interesting task is to apply a model for a specific task, such as the prediction of student's performance, for another related task, such as the prediction of student's dropout or for regression tasks (e.g., for predicting students' grades). In a future work we will also investigate the efficiency of transfer learning in imbalanced datasets obtained from several educational settings. If someone has only the target task, but also has the ability to choose a limited number of additional training data to collect, then active learning algorithms can be used to make choices that will improve the performance on the target task. These algorithms may also be combined into active transfer learning [33].

Author Contributions: Conceptualization, M.T. and S.K.; methodology, S.K.; software, M.T.; validation, G.K., S.K. and O.R.; formal analysis, M.T.; investigation, M.T.; resources, S.K.; data curation, G.K.; writing—original draft preparation, M.T.; writing—review and editing, G.K.; visualization, M.T.; supervision, S.K.; project administration, O.R.; funding acquisition, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pan, S.J.; Yang, Q. A survey on transfer learning. *Ieee Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
3. Brownlee, J. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*; Machine Learning Mastery: Vermont Victoria, Australia, 2019.
4. Brownlee, J. *Deep Learning with Python: Develop Deep Learning Models on Theano and Tensorflow Using Keras*; Machine Learning Mastery: Vermont Victoria, Australia, 2016.
5. Ng, A. Nuts and bolts of building AI applications using Deep Learning. Nips Keynote Talk. In Proceedings of the Thirtieth Conference on Neural Information Processing Systems. 2016 NIPS'16, Barcelona, Spain, 5–10 December 2016.
6. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014.
7. Liz-Domínguez, M.; Caeiro-Rodríguez, M.; Llamas-Nistal, M.; Mikic-Fonte, F.A. Systematic literature review of predictive analysis tools in higher education. *Appl. Sci.* **2019**, *9*, 5569. [CrossRef]
8. Boyer, S.; Veeramachaneni, K. Transfer learning for predictive models in massive open online courses. In *International Conference on Artificial Intelligence in Education*; Springer: Berlin/Heidelberg, Germany, 2015.
9. Ding, M.; Wang, Y.; Hemberg, E.; O'Reilly, U.-M. Transfer Learning using Representation Learning in Massive Open Online Courses. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge, Tempe, AZ, USA, 4–8 March 2019.
10. Boyer, S.A. Transfer Learning for Predictive Models in MOOCs. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2016.
11. Guo, B.; Zhang, R.; Xu, G.; Shi, C.; Yang, L. Predicting students performance in educational data mining. In Proceedings of the 2015 International Symposium on Educational Technology (ISET), Wuhan, China, 27–29 July 2015.
12. Okubo, F.; Yamashita, T.; Shimada, A.; Ogata, H. A neural network approach for students' performance prediction. In Proceedings of the Seventh International Learning Analytics & Knowledge Conference, Vancouver, BC, Canada, 13–17 March 2017.
13. Kim, B.-H.; Vizitei, E.; Ganapathi, V. GritNet: Student performance prediction with deep learning. *arXiv* **2018**, arXiv:1804.07405.

14. Kostopoulos, G.; Tsiakmaki, M.; Kotsiantis, S.; Ragos, O. Deep Dense Neural Network for Early Predic-tion of Failure-Prone Students. In *Machine Learning Paradigms-Advances in Theory and Applications of Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2019.
15. Wang, W.; Yu, H.; Miao, C. Deep model for dropout prediction in MOOCs. In Proceedings of the ACM International Conference Proceeding Series, Beijing, China, 6–9 July 2017.
16. Whitehill, J.; Mohan, K.; Seaton, D.; Rosen, Y.; Tingley, D. Delving Deeper into MOOC Student Dropout Prediction. *arXiv* **2017**, arXiv:1702.06404.
17. Xing, W.; Du, D. Dropout prediction in MOOCs: Using deep learning for personalized intervention. *J. Educ. Comput. Res.* **2018**, *57*, 547–570. [CrossRef]
18. Bosch, N.; Paquette, L. Unsupervised Deep Autoencoders for Feature Extraction with Educational Data. In Proceedings of the Deep Learning with Educational Data Workshop at the 10th International Conference on Educational Data Mining, Wuhan, Hubei, 25–28 June 2017.
19. Ruder, S.; Peters, M.E.; Swayamdipta, S.; Wolf, T. Transfer learning in natural language processing. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, Minneapolis, MN, USA, 2 June 2019.
20. Weiss, K.; Khoshgoftaar, T.M.; Wang, D.D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1345–1459. [CrossRef]
21. Arnold, A.; Nallapati, R.; Cohen, W.W. A Comparative Study of Methods for Transductive Transfer Learning. In Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, Omaha, NE, USA, 28–31 October 2007.
22. Romero, C.; Ventura, S. Educational data mining and learning analytics: An updated survey. *Wiley Interdiscip. Rev.* **2020**, e1355. [CrossRef]
23. Moreno-Marcos, P.M.; Alario-Hoyos, C.; Muñoz-Merino, P.J.; Kloos, C.D. Prediction in MOOCs: A review and future research directions. *IEEE Trans. Learn. Technol.* **2018**, *12*, 384–401. [CrossRef]
24. Costa, E.B.; Fonseca, B.; Santana, M.A.; de Araújo, F.F.; Rego, J. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Comput. Hum. Behav.* **2017**, *73*, 247–256. [CrossRef]
25. Vitiello, M.; Walk, S.; Chang, V.; Hernandez, R.; Helic, D.; Guetl, C. MOOC dropouts: A multi-system classifier. In Proceedings of the European Conference on Technology Enhanced Learning, Tallinn, Estonia, 12–15 September 2017.
26. Hunt, X.J.; Kabul, I.K.; Silva, J. Transfer Learning for Education Data. In Proceedings of the KDD Workshop, Halifax, NS, Canada, 13–17 August 2017.
27. Tri, P.T.; Chau, V.T.N.; Phung, N.H. Combining transfer learning and case-based reasoning for an educational decision making support model. In Proceedings of the Multi-disciplinary Trends in Artificial Intelligence: 11th International Workshop, MIWAI 2017, Gadong, Brunei, 20–22 November 2017.
28. Zeng, Z.; Chaturvedi, S.; Bhat, S.; Roth, D. DiAd: Domain adaptation for learning at scale. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge, Tempe, Arizona, 4–8 March 2019.
29. López-Zambrano, J.; Lara, J.A.; Romero, C. Towards Portability of Models for Predicting Students' Final Performance in University Courses Starting from Moodle Logs. *Appl. Sci.* **2020**, *10*, 354. [CrossRef]
30. Tsiakmaki, M.; Kostopoulos, G.; Kotsiantis, S.; Ragos, O. Implementing AutoML in Educational Data Mining for Prediction Tasks. *Appl. Sci.* **2019**, *10*, 90. [CrossRef]
31. Chollet, F. Keras. Available online: https://keras.io (accessed on 1 January 2020).
32. Romero, C.; Ventura, S. Data mining in education. *Wiley Interdiscip. Rev.* **2013**, *3*, 12–27. [CrossRef]
33. Wang, X.; Huang, T.-K.; Schneider, J. Active transfer learning under model shift. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014.

applied sciences

Article

Analysis of Cross-Referencing Artificial Intelligence Topics Based on Sentence Modeling

Hosung Woo [1], JaMee Kim [2] and WonGyu Lee [3,*]

[1] Department of Computer Science and Engineering, Graduate School, Korea University, Seoul 02841, Korea; hosung.woo@inc.korea.ac.kr
[2] Major of Computer Science Education, Graduate School of Education, Korea University, Seoul 02841, Korea; celine@korea.ac.kr
[3] Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul 02841, Korea
* Correspondence: lee@inc.korea.ac.kr

Received: 5 May 2020; Accepted: 25 May 2020; Published: 26 May 2020

Abstract: Artificial intelligence (AI) is bringing about enormous changes in everyday life and today's society. Interest in AI is continuously increasing as many countries are creating new AI-related degrees, short-term intensive courses, and secondary school programs. This study was conducted with the aim of identifying the interrelationships among topics based on the understanding of various bodies of knowledge and to provide a foundation for topic compositions to construct an academic body of knowledge of AI. To this end, machine learning-based sentence similarity measurement models used in machine translation, chatbots, and document summarization were applied to the body of knowledge of AI. Consequently, several similar topics related to agent designing in AI, such as algorithm complexity, discrete structures, fundamentals of software development, and parallel and distributed computing were identified. The results of this study provide the knowledge necessary to cultivate talent by identifying relationships with other fields in the edutech field.

Keywords: Machine learning analysis; sentence modeling; topic analysis; cross referencing topic

1. Introduction

Information technology (IT) is driving changes in society and leading to a new paradigm shift in national development across the globe. Among them, artificial intelligence (AI)-related research and talent cultivation are becoming the basis for national development, while emerging as a competitive edge. Through AI.gov, the United States has promoted AI research and development at the government level since February 2019 [1], and at the Future Strategy Innovation Conference in March 2019, Japan presented AI, quantization, and biotechnology as three strategic technologies that will lead Japan's development [2]. South Korea's Ministry of Science and ICT also announced its plans to foster AI talent by 2022 according to the "AI Research & Development Strategy" of May 2018 [3].

These interests in AI from different countries are motivating talent cultivation. This is seen in the creation of AI degree programs at universities, the dedication of new colleges to AI, the design of new AI degree programs, etc. The U.S. has already implemented talent development plans at universities [4,5], while Japan has introduced its future strategic plans through linking AI with primary and secondary education. China, through the "New Generation Artificial Intelligence Development Plan" and the "AI Innovation Action Plan" of 2017 and 2018, has begun AI-focused primary and secondary education where students study the core technologies of AI [6,7]. India also announced an AI curriculum inclusion for eighth and ninth graders at the 2019 Central Board of Secondary Education [8].

As a way of implementing various AI professional policies, AI is being taught at the primary school level; however, to date, no core knowledge has been defined regarding AI education. In the absence of these core content standards, the structures of AI education vary among different educational institutions [9,10]. In a similar context, the British Prime Minister emphasized the importance of establishing rules and standards for AI technologies at the 2018 World Economic Forum. Although AI is now in the spotlight, both academically and economically, it is widely regarded as a union of knowledge containing topics of various fields. Therefore, developing a body of knowledge in the field of AI will also help in establishing an academic foundation for AI.

A body of knowledge is a reconstruction of the knowledge area (KA) based on the knowledge that experts in academic fields must obtain. It is important for the topics covered in the knowledge areas to be constructed in a way that they correlate with other areas or topics. This means that the body of knowledge based on semantic relations can be said to correspond to the overall knowledge that one must acquire in the given academic field.

Therefore, this study focused on deriving areas and topics that correlate with the field of AI. To fulfill its goals, this study derived knowledge areas and topics using sentence models. AI's body of knowledge extracted from sentence models will provide implications for which research topics should be continued from the academic perspective. In particular, by identifying relationships with other areas, it will also contribute to constructing the knowledge that is required to develop professionals at the university level.

2. Related Research

A body of knowledge analysis aims to identify the hierarchies of knowledge areas or relationships among topics. In some cases, a visual representation is used for a clear knowledge sequence or secondary utilization of knowledge. Analysis and reorganization of a body of knowledge should either be performed by experts or computer systems. Using an expert can be expensive and time-consuming, and maintaining consistency is quite challenging. Therefore, computer systems are used instead. This section discusses previous studies on body of knowledge analysis and sentence modeling using computer systems.

2.1. Sentence Modeling

Analyzing the body of knowledge for each academic field takes curriculum composition or evaluation into account. Various types of research related to body of knowledge analysis have been conducted. The main types are discussed below.

The first type of body of knowledge analysis research is category-based research. Based on knowledge about the units of the Computing Curricula 2001, which can be regarded as the computer science (CS) body of knowledge, research was performed to set the flow of the teaching syllabus. A syllabus-maker that can develop or analyze a teaching syllabus has been proposed [11]. By using the body of knowledge of Computer Science Curricula 2013 (CS2013), the distribution of knowledge areas for a teaching syllabus has also been predicted [12]. This combined learners with learner information in an attempt to provide them with personalized learning paths. Further, Ida (2009) proposed a new analysis method, called the library classification system, that is based on the classification information in the curriculum's body of knowledge, instead of the body of knowledge.

In the abovementioned studies, analysis was performed based on categories such as KA and knowledge units (KU), which do not take specific topics or the contents of each body of knowledge into account. Hence, although they can be useful in identifying the entire frame or hierarchy of a body of knowledge, they have a limited ability to identify meaning based on the detailed contents of the subject unit.

The second type of body of knowledge analysis research is word-based research, which uses stochastic topic modeling. Topic modeling estimates topics based on the distribution of words contained in a document. Latent Dirichlet allocation (LDA) is one of the popular methods used in topic modeling [13]. Sekiya (2014) used supervised LDA (sLDA) to analyze the changes in the body of knowledge of CS2008 and CS2013 [14]. More specifically, ten words that could represent each KA in the body of knowledge were extracted and compared based on common words. Another study was based on the CS2013 body of knowledge, which applied an extended method called the simplified, supervised latent Dirichlet allocation (ssLDA) [15]. Syllabus topics from the top 47 universities worldwide in 2014–2015 were quantified. Subsequently, based on the related topics, features of similar topics were analyzed.

In 2018, the Information Processing Society of Japan presented the computer science body of knowledge with 1540 topics. The presented topics ranged from approximately one to five words. For example, if the topic called "Finite-state Machine" is processed on a word basis, the result would include various machine-related topics such as Turing machines, assembly level machines, and machine learning. This means that it is difficult to extract the exact meaning from the given topic. As with previous studies, unigram-based research through topic modeling is suitable for classifying words that appear in the body of knowledge by topic, but it can be semantically limited as it treats topics segmentally.

For analysis based on an accurate understanding of the body of knowledge, it is necessary to apply unprocessed topics. This would help in providing a clear meaning of the topics as well as to guess the relationships among the words. Therefore, this study uses sentence modeling to find topic meanings in the body of knowledge and to identify the relationships among these topics.

2.2. *Knowledge Areas Analysis Research*

Sentence modeling is an important problem in natural language processing [16]. It allows the insertion of sentences into vector spaces and uses the resulting vectors for classification and text generation [17]. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are generally used in sentence modeling approaches.

CNNs that save local information about sentences use filters, which are applied to local characteristics on the layers [18]. Figure 1 depicts a simple CNN structure with a convolutional layer on top of a word vector obtained from an unsupervised neural language model. This model achieved several excellent benchmark results even without parameter tuning [19].

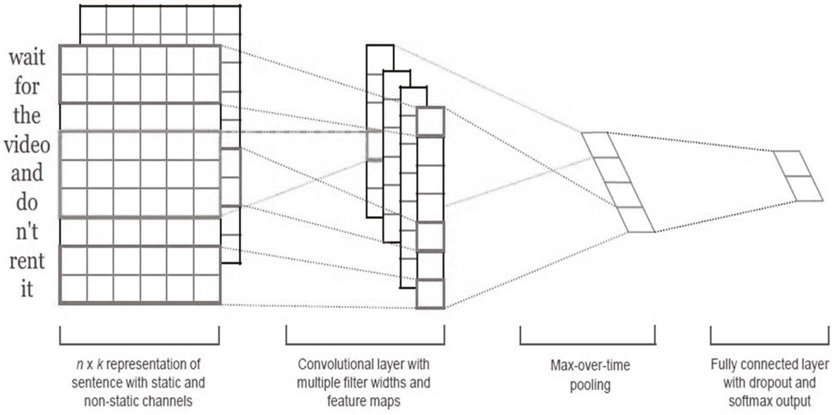

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1. Structure of convolutional neural networks (CNN).

The CNN models used in early computer vision are also effective in natural language processing, and they have been applied in semantic parsing, search query retrieval [20], sentence modeling [21], and other traditional natural language processing (NLP) works [22].

RNNs process word inputs in a particular order and learn from the order of appearance of particular expressions. Thus, modeling can process semantic similarities between sentences and phrases. RNNs can also use their own models, but they are used as an extension of the modified model. Manhattan long short-term memory (LSTM), which is a modified RNN model, uses two LSTMs to read and process word vectors that represent two input sentences. This model has been shown to outperform the complex neural network model. The structure of the Manhattan LSTM is shown in Figure 2 [23]. The main feature of this model is that LSTM and the Manhattan metric are used based on a Siamese network structure that contains two identical subnetwork components. The hidden state h1 is learned through word vectors and randomly generated weights. Then, the hidden state sentence ht is generated based on the input function using hidden state ht-1 and position t. Subsequently, the semantic similarity of sentences is measured using the vectors of the final hidden state. For example, "he," "is," and "smart" are the words in the vectors xi; x1 is the input vector of h1 and is used to calculate the status value of h1; h2 is calculated by referring to the previous state value and x2; and the final hidden state, h3, is calculated through x3 and the previous state value h2. "a," "truly," "wise," and "man" are treated in the same manner, and similarities are measured with the final vectors calculated by processing two sentences.

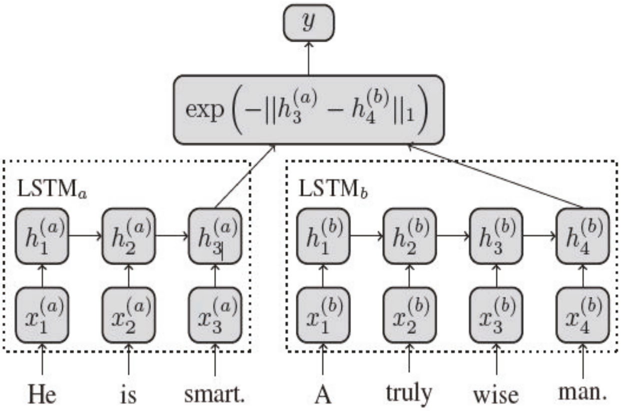


Figure 2. Structure of Manhattan LSTM.

In addition to the abovementioned models, modified RNN models such as bidirectional LSTM, multidimensional LSTM [24], gated recurrent unit [25], and recursive tree structure [26] have been applied to text modeling through the existing model architectural modifications. RNN-based models are generally calculated from the word order of the input and output sentences. Sequential processing in the model learning process can reflect the syntax and semantics of the given sentences, but it has a slow computational speed and parallelism, which is a limitation.

There are also structures that allow modeling dependencies regardless of the length of the input or output sentences for computation efficiency. Figure 3 shows the structure of the transformer model, which is based only on attention mechanisms without using CNN or RNN [27]. Attention mechanisms refer back to the entire input sentence of the encoder at every time-step of the output word prediction from the decoder. At this stage, the entire input sentence is not considered. Instead, the focus is on the input words at a specific time based on the similarity. The transformer model allows parallelism, and it can emphasize the values that are the most closely related to the query through "attention" in the encoder and decoder.

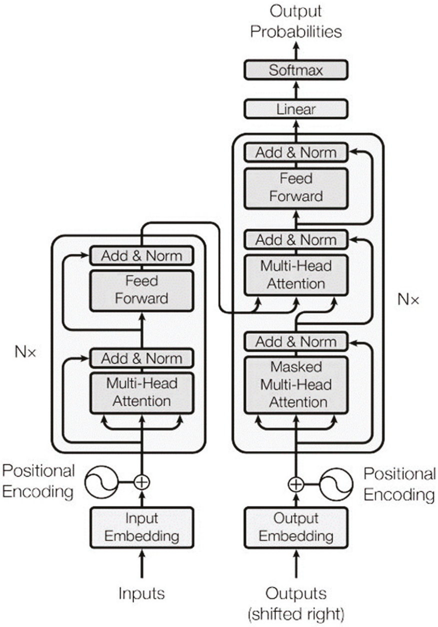


Figure 3. Structure of the transformer model.

The CNN, Manhattan LSTM, and transformer models complement each other's strengths and weaknesses. Therefore, it is essential to implement all three models to evaluate the semantic similarity of body of knowledge topics and subsequently select the model with the highest accuracy for conducting cross-referencing among the topics.

3. Methods

3.1. Experimental Procedure

AI is an artificial implementation of human intelligence, partially or fully, based on the broad concept of "smart" computers. Intelligent systems (IS) operate in the same manner as AI, but they do not feature deep neural networks that support self-learning [28]. In the field of CS, IS contain AI-related contents. In this study, an IS was used to derive the body of knowledge of AI. The procedure used to derive cross-references among topics presented in the IS knowledge area of the CS field was as follows.

Step 1. From the CS2013 body of knowledge, the topics of IS and other areas were classified.

Step 2. To calculate the similarity among topics, three different sentence models (CNN, MaLSTM, and transformer) based on machine learning were implemented. This system was developed using Python 3.6 and executed on Linux 16.04.

Step 3. The models were trained using data from Stanford Natural Language Inference (SNLI) corpus and Quora Question Pairs (QQP).

Step 4. The accuracy was calculated after training, and the sentence model with the highest accuracy was selected.

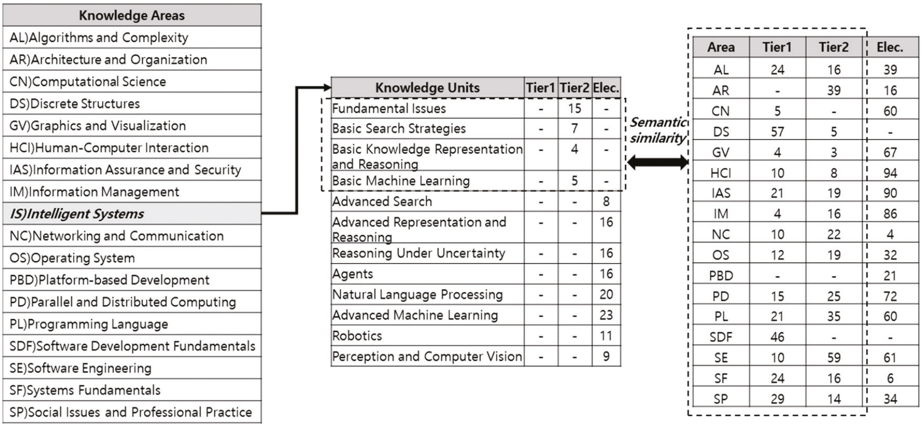
Step 5. Semantic similarities between classified IS topics and topics from other areas were calculated.

Step 6. Through various similarity simulations between the two topics, the similarity levels were divided into "0.95 < similarity" and "0.90 < similarity ≤ 0.95."

Step 7. A search engine was used to examine the semantic validity of the topics with a similarity of "0.95 < similarity."

3.2. Subject of Analysis

"Computer Science Curricula CS2013," which is a standard body of knowledge of the CS field presented by ACM and IEEE Computer Association, was selected. The body of knowledge CS2013 comprises eighteen KA, and each KA contains approximately ten KUs, as shown in Figure 4. Each KU subject has three tiers: tier 1 covers the basic introductory concepts, tier 2 covers the undergraduate-major concepts, and the elective tier covers the contents that are more advanced than the undergraduate-major contents.



Knowledge Areas
AL)Algorithms and Complexity
AR)Architecture and Organization
CN)Computational Science
DS)Discrete Structures
GV)Graphics and Visualization
HCI)Human-Computer Interaction
IAS)Information Assurance and Security
IM)Information Management
IS)Intelligent Systems
NC)Networking and Communication
OS)Operating System
PBD)Platform-based Development
PD)Parallel and Distributed Computing
PL)Programming Language
SDF)Software Development Fundamentals
SE)Software Engineering
SF)Systems Fundamentals
SP)Social Issues and Professional Practice

Knowledge Units	Tier1	Tier2	Elec.
Fundamental Issues	-	15	-
Basic Search Strategies	-	7	-
Basic Knowledge Representation and Reasoning	-	4	-
Basic Machine Learning	-	5	-
Advanced Search	-	-	8
Advanced Representation and Reasoning	-	-	16
Reasoning Under Uncertainty	-	-	16
Agents	-	-	16
Natural Language Processing	-	-	20
Advanced Machine Learning	-	-	23
Robotics	-	-	11
Perception and Computer Vision	-	-	9

Area	Tier1	Tier2	Elec.
AL	24	16	39
AR	-	39	16
CN	5	-	60
DS	57	5	-
GV	4	3	67
HCI	10	8	94
IAS	21	19	90
IM	4	16	86
NC	10	22	4
OS	12	19	32
PBD	-	-	21
PD	15	25	72
PL	21	35	60
SDF	46	-	-
SE	10	59	61
SF	24	16	6
SP	29	14	34

Figure 4. Computer Science Curricula CS2013 body of knowledge.

To derive cross-references among the various topics in the body of knowledge of AI, this study was conducted based on the topics of IS, and each topic was classified as follows. First, the CS2013 body of knowledge was divided into knowledge areas, knowledge units, and topics. Second, CS2013 classified topics of KU into Tier 1, Tier 2, and elective for all eighteen KAs. Third, the IS consisted of four units of knowledge: fundamental issues, basic search strategies, basic knowledge representation and reasoning, and basic machine learning. All four KUs are of the Tier 2 level with 31 topics. Fourth, 17 areas excluding IS consisted of 323 Tier 1 and 327 Tier 2 topics. Thus, the similarities among the 31 topics of IS and 660 topics of 17 KUs were analyzed based on levels.

3.3. Sentence Model Performance

CNN, Manhattan LSTM, and multi-head attention networks (a transformer model) were implemented, and their performances were compared. SNLI and QQP were used as corpuses for model training. For each corpus, 90% of the data was used as the training set, and 10% was used as the testing set to verify the accuracy of the model.

- Training set: SNLI number: 330,635, QQP number: 363,861
- Testing set: SNLI number: 36,738, QQP number: 40,430

The content and accuracy of the models are detailed in Table 1.

In QQP, CNN had an accuracy of 83.8%, which was superior to Manhattan LSTM's 82.8% accuracy. Conversely, Manhattan LSTM had an accuracy of 80.6% in SNLI, which was the highest accuracy among the three models. Multi-head attention networks showed better accuracy than CNN in SNLI, but the mean accuracy was the lowest among all. In this study, Manhattan LSTM was used, and it showed the highest accuracy when the model accuracy for both corpuses were converted to average.

Table 1. Content and accuracy of sentence models.

Sentence Model	Configuration	Accuracy		Mean
		SNLI	QQP	
Convolutional Neural Networks	Number of filters: 50, 50, 50 Filter size: 2, 3, 4	0.783	0.838	0.81
Manhattan LSTM	Hidden size: 128 Cell type: GUR	0.806	0.828	0.817
Multi-head Attention Networks	Number of blocks: 2 Number of heads: 8 Layers normalization: False	0.802	0.809	0.805

3.4. Setting the Similarity of the Sentence Model

This study was performed based on the Manhattan LSTM, which had the highest average accuracy of the three models used. In the two sentences, the semantic similarity level in the vectors of the final hidden state was modified to provide outputs according to the threshold values, as shown in Figure 5.

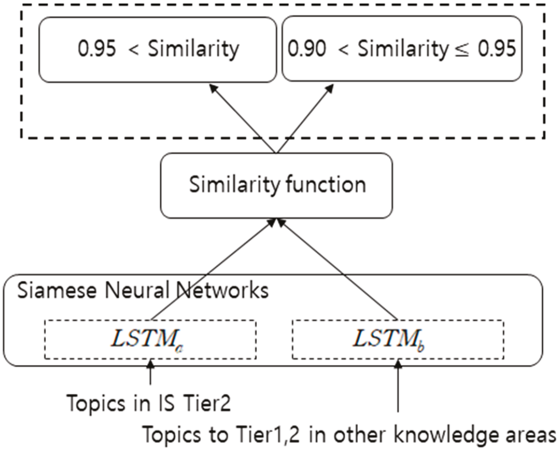


Figure 5. Structure of the sentence model used.

The threshold value was set at two levels through various experiments. In other words, the similarity between topics is either greater than 0.95 or just 0.9 but less than or equal to 0.95.

Manhattan distance was applied as a similarity function. In general, the Euclidean distance is not used in the problem of determining similarity because it causes learning to be slow and it has difficulty correcting errors owing to vanishing gradients in the early stage of learning [29]. On the other hand, Manhattan distance can match values so that two sentences are close to one if they are semantically similar and close to zero otherwise, without a separate activation function to determine the output of the neural network in the form of e^{-x}. The cost function uses MSE for this difference.

4. Application Results

4.1. Sentence Model Performance

Table 2 presents the results of the analysis of the 31 topics of IS and 323 topics in other areas. More semantically similar topics were extracted as having "0.90 < similarity ≤ 0.95" than "0.95 < similarity." This was due to "0.95 < similarity" being more robust than "0.90 < similarity ≤ 0.95."

Table 2. Tier 1: Topic pairs with high similarity between IS and topics in other areas.

KA	Topic Number of KA	0.95 < Similarity	0.90 < Similarity ≤ 0.95
AL	24	2	17
AR	0	0	0
CN	5	0	0
DS	57	1	74
GV	4	0	0
HCI	10	0	0
IAS	21	0	7
IM	4	0	0
NC	10	1	4
OS	12	0	1
PBD	0	0	0
PD	15	0	6
PL	21	0	4
SDF	46	1	31
SE	10	0	1
SF	24	0	9
SP	29	1	8

There were 74 pairs extracted from 57 topics of DS with similarity range of "0.90 < similarity ≤ 0.95." In addition, 31 topics of IS and 57 topics of DS were compared as 1,767 (31 × 57) topic pairs. Next, many topic pairs with high similarity appeared in the order of SDF and AL. For "0.95 < similarity," six pairs were extracted from five areas; namely, AL, DS, NC, SDF, and SP. AL had two pairs while the other four areas had a pair each. The details are shown in Figure 6.

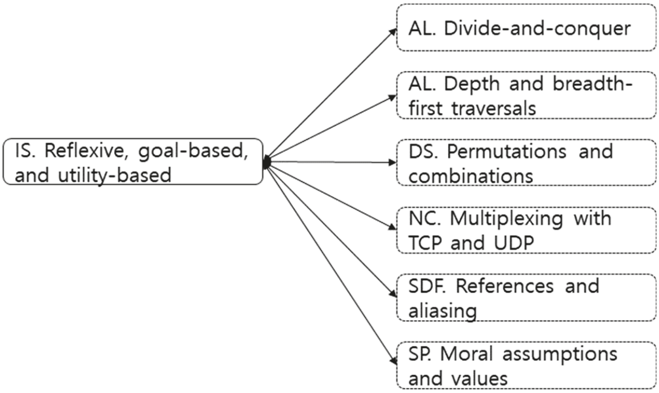

IS. Reflexive, goal-based, and utility-based
AL. Divide-and-conquer
AL. Depth and breadth-first traversals
DS. Permutations and combinations
NC. Multiplexing with TCP and UDP
SDF. References and aliasing
SP. Moral assumptions and values

Figure 6. "0.95 < similarity": Topics with high relevance in Tier 1.

The topic pairs with "0.95 < similarity" were focused upon because a high similarity is shown in a robust state. In other words, it was determined that knowledge could be extracted in the order of higher similarity with IS first.

We now examine "Reflexive, goal-based, and utility-based" from the perspective of AI. An agent of AI is an autonomous process that automatically performs tasks for users. It is one of the software systems with independent functions that perform tasks and typically operate in a distributed environment [30]. To perform tasks, an agent interacts with different agents through its own reasoning method using a database of non-procedural processing information called knowledge. Then, the agent continues to act based on the learning and purpose-oriented skills obtained from the experiences. This means that "Reflexive, goal-based, and utility-based" explains how to design an agent program. In this study, as shown in Figure 6, the following were included in the knowledge to be handled before learning the agent.

First, agents interact with external environments using sensors to achieve their goals in the complex and fluid real-world environments. In other words, agents can be said to be faced with a complex problem that requires considering various situations. Divide-and-conquer strategies are effective in solving complex problems. With agent designs, it is easy and simple to approach and solve several smaller sub-problems [31]. This problem-solving strategy can be applied not only to agent design but also to general problem-solving.

Second, to achieve a predefined goal or solution, a systematic search method is applied for problem-solving in the agents "depth and breadth-first traversals," which is a search algorithm in a common search strategy [32].

Third, frames are a means of expressing knowledge in the agents. A knowledge consists of small packets called frames where the contents of a frame are specific slots with values. A topic, "permutations and combinations," is a basic concept that helps with finding or identifying new knowledge [33].

Fourth, there is a need to understand the "references and aliasing" topic to recognize and reason knowledge in agents. There are physical difficulties such as memory limits in perceiving all the knowledge in a computing environment. To solve this problem, knowledge can be limited to certain categories and then perceived. Processing can then be done by referencing the address of the memory where the knowledge is stored [34].

Fifth, the main rationale for constructing multi-agent systems is that by forming a community, multiple agents can provide more added values than one agent can provide. Additionally, agents can participate in an agent society through communications, and they can acquire services owned by other agents through interactions. In other words, even if an agent does not have all the information, it can provide various services because of its interactions with other agents [35]. The concept of "multiplexing with TCP and UDP" has been applied to implement the means of information exchange and communication in the agent society.

Finally, the progress and globalization of various technologies including agents are affecting society. Ethical and moral issues that can arise from agents are indispensable factors in terms of education. "Moral assumptions and values" can be said to be a required topic for the development and use of technology [36].

The topics that have semantic similarities with IS at the "0.90 < similarity ≤ 0.95" range are shown in Table 3.

The areas with the highest distribution of similar topics were in the order of DS (Discrete Structures), SDF (Software Development Fundamentals), and AL (Algorithms and Complexity). Topics that were extracted from the DS included topics covered by IS, such as mathematical proofs and development of the ability to understand concepts [37]. In other words, these areas include important contents such as set theory, logic, graph theory, and probability theory, which are foundations of computer science.

Students should be able to implement IS to solve problems effectively in the field of AI. This means that they should be able to read and write programs in several programming languages. Topics that were sampled from SDF were more than just programming skill topics. They included basic concepts and techniques in the software development process, such as algorithm design and analysis, proper paradigm selection, and the use of modern development and test tools.

Performance may vary in implementing IS depending on both the accuracy of the model trained with a large amount of data, and the chosen algorithm and its suitability. In other words, algorithm design is very important to improve the efficiency of the IS model design [31]. AL is said to be the basis of computer science and software engineering as well as IS design. As demonstrated by brute-force algorithms, greedy algorithms, and search algorithms, building an understanding and insight into the algorithms is one of the important factors in IS composition. As seen from the results of the study, the topics with high similarity which were sampled from this study are closely related to the IS and AI fields.

Table 3. "0.90 < similarity ≤ 0.95": Topics with high similarity in Tier 1.

KA	0.90 **< Similarity** ≤ 0.95	
AL	Brute-force algorithms Greedy algorithms Recursive backtracking	Dynamic programming Sequential and binary search algorithms
DS	Sets Relations Functions Surjections, injections, bijections Inverses Composition Logical connectives Predicate logic Direct proofs Disproving by counterexample Proof by contradiction Structural induction Counting arguments Inclusion-exclusion principle Basic definitions Properties	Traversal strategies Undirected graphs Directed graphs Weighted graphs Conditional probability, Bayes' theorem Expectation, including linearity of expectation Cardinality of finite sets Propositional inference rules (concepts of modus ponens and modus tollens) Universal and existential quantification Weak and strong induction (i.e., first and second principle of induction) The pigeonhole principle The binomial theorem
IAS	CIA (Confidentiality, Integrity, Availability) XSS vulnerability	Input validation and data sanitization
NC	Layering principles (encapsulation, multiplexing)	
OS	Design issues (efficiency, robustness, flexibility, portability, security, compatibility)	
PD	Shared Memory	Multicore processors
PL	A type as a set of values together with a set of operations	Effect-free programming
SDF	Problem-solving strategies Divide-and-conquer strategies Abstraction Arrays Queues Sets Maps	Simple refactoring Debugging strategies Documentation and program style Iterative and recursive traversal of data structures Abstract data types and their implementation
SE	Programming in the large vs. individual programming	
SF	Reliability Combinational logic, sequential logic, registers, memories	Parallel programming vs. concurrent programming Request parallelism vs. task parallelism Application-virtual machine interaction
SP	Ethical argumentation	Plagiarism

4.2. Results for Tier 2

Table 4 presents the results of analyzing 327 Tier 2 topics in the IS and other areas. As shown in Table 2, Table 4 also shows more similar topic samples at the "0.90 < similarity ≤ 0.95" range than at the "0.95 < similarity" level. At the "0.90 < similarity ≤ 0.95" range, SE had 31 topic pairs of samples from 59 topics. The next highest area was PD (Parallel and Distributed Computing). At the "0.95 < similarity" level, ten pairs were extracted from seven areas, and the NC (Networking and Communication) topic had the most pairs compared to other topics with three pairs. As shown in Figure 7, the three topics of IS and the topic of "Routing versus forwarding" of NC were similar.

Table 4. Tier 2: Topic pairs with high similarity between IS and other topic areas.

KA	Topic Number of KA	0.95 < Similarity	0.90 < Similarity ≤ 0.95
AL	16	1	17
AR	39	2	9
CN	0	0	0
DS	5	0	6
GV	3	0	0
HCI	8	0	1
IAS	19	0	1
IM	16	1	5
NC	22	3	15
OS	19	0	11
PBD	0	0	0
PD	25	1	26
PL	35	0	0
SDF	0	0	0
SE	59	1	31
SF	16	0	1
SP	14	1	8

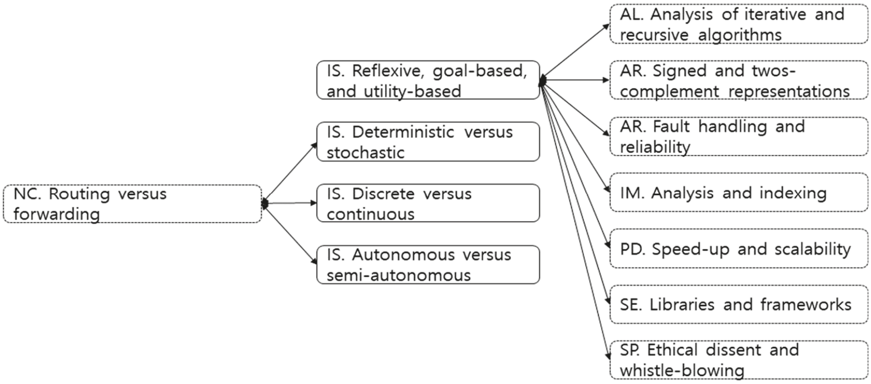

AL. Analysis of iterative and recursive algorithms
IS. Reflexive, goal-based, and utility-based
AR. Signed and twos-complement representations
IS. Deterministic versus stochastic
AR. Fault handling and reliability
NC. Routing versus forwarding
IS. Discrete versus continuous
IM. Analysis and indexing
IS. Autonomous versus semi-autonomous
PD. Speed-up and scalability
SE. Libraries and frameworks
SP. Ethical dissent and whistle-blowing

Figure 7. "0.95 < Similarity": Topics with high relevance in Tier 2.

Based on CS2013, Tier 2 features more advanced topics compared to Tier 1. To learn these topics, results from Section 4.1 can be considered as prerequisite knowledge or references.

As shown in Figure 7, seven topic pairs from six areas were extracted from "reflexive, goal-based, and utility-based" of the "0.95 < similarity" level. AR (Architecture and Organization) had two pairs, while the other five areas had a topic pair each.

Looking at the AL around "reflexive, goal-based, and utility-based," Tier 1 is about problem-solving and graph-based querying (see Figure 6). On the other hand, Tier 2 is about an approach to algorithm recursion or iteration with "analysis of iterative and recursive algorithms." In other words, the similar topics of Tier 1 are related to theories or concepts for problem-solving, whereas Tier 2 is related to strategies for efficiently implementing agents [37].

Stability is an important factor for processing the knowledge of agents and for agents to interact with one another using processed knowledge. In terms of implementation, "signed and two's complement representations" is related to the sign and representation range variables to be saved in the memory. Memory overflow occurs when the data overrun the designated boundary of the memory space. This can lead to unexpected behaviors or computer security vulnerabilities [38]. In addition, "fault handling and reliability" is also about ensuring stability by handling the fault of the memory system.

In addition to the abovementioned, concepts related to extending or parallelizing the information system of agents and the topics related to tools that can be utilized without having to implement everything from scratch were extracted. Agents, which are typically complex and massive systems, are developed in a collaborative design environment. "Ethical dissent and whistle-blowing," which is an ethical and altruistic act, is about helping to improve the development of an organization and further contribute to forming a healthy community [39].

"Deterministic versus stochastic," "discrete versus continuous," and "autonomous versus semi-autonomous" were very similar to "routing versus forwarding" of NC. The three topics of IS were related to the characteristics of the given problems, which needed to be optimally inferred or solved by the agents. These concepts are applied in various fields in addition to AI. This means that it is an important topic for agents. However, the importance can also be applied to routing algorithms for deterministic routing and probabilistic routing [40], autonomous systems for network topology management and control [41], and the process of finding a network [42].

Table 5 presents topics with similarity to IS in Tier 2 at the "0.90 < Similarity ≤ 0.95" range. Topics were extracted from 13 of 17 areas excluding IS. Of these, the SE (Software Engineering) and PD (Parallel and Distributed Computing) areas had the most topics extracted. Schedule, cost, and quality are important factors in the design, implementation, and testing phases of IS. The SE topics are applicable to software development in all areas of computing, including IS [43]. SE, along with IS, can be said to be related to the application of theory, knowledge, and practice to build general-purpose software more efficiently.

Table 5. "0.90 < similarity ≤ 0.95": Topics with a high similarity in Tier 2.

KA	0.90 **< Similarity** ≤ 0.95	
AL	Heuristics Heaps	Context-free grammar
AR	Instruction formats RAID architectures	Multimedia support
DS	Graph isomorphism	Variance
HCI	Low-fidelity (paper) prototyping	
IAS	Use of vetted security components	
IM	Declarative and navigational queries, use of links	
NC	Multiple Access Problem TCP Ethernet Switching	Fixed allocation (TDM, FDM, WDM) versus dynamic allocation Need for resource allocation Fairness
OS	Backups Processes and threads Multiprocessor issues (spin-locks, reentrancy)	Caching Deadlines and real-time issues Schedulers and policies
PD	Task-based decomposition Data-parallel decomposition Composition Symmetric multiprocessing (SMP)	Naturally (embarrassingly) parallel algorithms Implementation strategies such as threads Message buffering Message passing
PL	Definition	
SE	Continuous integration Software requirements elicitation Integration strategies Testing fundamentals Refactoring Software evolution Software reuse Components	Software configuration management and version control Tool integration concepts and mechanisms Evaluation and use of requirements specifications Product lines Non-functional requirements and their relationship to software quality
SF	Redundancy through check and retry	
SP	Context-aware computing Forms of professional credentialing	Accessibility issues, including legal requirements

In terms of the efficiency of IS, understanding parallel algorithms, strategies for problem decomposition, system architectures, detailed implementation strategy, and performance analysis and tuning is important [31]. For this reason, the topics extracted from PD, such as concurrency and parallel execution, consistency of status and memory operation, and latency, had high similarities.

4.3. KU of IS vs. KU of Other Knowledge Areas

The following four units were the Tier 2 topics of academic major level in the IS areas of "fundamental issues," "basic search strategies," "basic knowledge representation and reasoning," and "basic machine learning." KUs with high similarity to the KUs of IS are shown in Table 6 below. At "0.95 < similarity," the KU topic "fundamental issues" was extracted. At "0.90 < similarity ≤ 0.95," all three KU topics except for "basic knowledge representation and reasoning" were extracted.

Out of 18 KAs, 53 KUs of 14 KAs showed a high similarity. The areas with high similarities to IS and KU were DS, NC, and SE. "Sets, relations, and functions" of DS had the highest similarity with the 14 KUs. If the topics were limited to KUs with more than five pairs of similar areas, "fundamental issues" was related to the 15 KUs. There were nine KUs corresponding to AL, DS, and SDF of Tier 1, and six KUs corresponding to AL, NC, PD, and SE of Tier 2. "Basic machine learning" had a high similarity with "algorithmic strategies" of AL, and "graphs and tree" and "sets, relations, and functions" of DS from Tier 1. "Fundamental issues," which covers the general contents of AI relative to the other KUs, had 111 pairs from Tier 1 and 84 pairs from Tier 2 with a high similarity. There were more than 40 pairs of topics on "basic machine learning," which covers the basics of machine learning, such as supervised learning, unsupervised learning, and reinforcement learning.

4.4. Evaluation

Currently, there is no research available on the curriculum relevance of the topics. The validity of the study was evaluated using two methods because a direct comparison with previous studies was not possible. The first method was a content validation from experts. The degree of similarity among the topics for content validity verification was quantified. At the same time, "rater reliability" among experts was determined. The second method was an index term-based method using a search engine. Specifically, the index term-based method was applied in the study of variables affecting similarities between two documents or two topics. This is because if the two topics appear simultaneously in the same document referring to a specific area, they can be said to be semantically related.

4.4.1. Content Validation by Experts

This study analyzed opinions from five experts based on the topics related to "reflexive, goal-based, and utility-based" for intelligent agent design. The experts met at least three of the following four criteria: Ph.D. in AI, seven years of work experience in the field of AI, seven years of AI research experience, and five years of experience teaching AI at a university.

The content validation proceeded as follows. In the first step, 66 topics were selected as random samples from four different similarity ranges: "0.95 < similarity," "0.95 ≥ similarity > 0.9," "0.9 ≥ similarity > 0.8," and "0.8 ≥ similarity > 0.7." In the second step, experts were asked to examine the extracted topics based on semantic relevance with "reflexive, goal-based, and utility-based" and the need for those topics. The results of experts on the relevance and need of each range of topics are shown in the Table 7.

Regarding the relevance of "reflexive, goal-based, and utility-based" to topics in "0.95 < similarity" range, the expert response was close to four at 3.74, suggesting a high "relevance." The necessity score of the topics for intelligent agent design was also high at 3.74.

For the topics in "0.95 ≥ similarity > 0.9" range, the relevance score was 3.49, and the "0.9 ≥ similarity > 0.8" range received a relevance score of 3.08. For the topics in the "0.8 ≥ similarity > 0.7" range, an "average" relevance score of 3.07 was received. In other words, based on the topics that were sampled in the study, it was shown that the higher the "similarity" range, the higher the relevance was for expert content validation. The "rater reliability" of experts was shown to be high at 0.90–0.78. Therefore, it can be concluded that topics sampled in the study for each range have a high semantic relevance.

Table 6. KU with high similarity to the KUs of IS.

KA.KU (Knowledge Area. Knowledge Unit)		**0.95 < Similarity**				**0.90 < Similarity ≤ 0.95**			
		Fundamental Issues		**Fundamental Issues**		**Basic Search Strategies**		**Basic Machine Learning**	
		Tier 1	**Tier 2**	**Tier 1**	**Tier 2**	**Tier 1**	**Tier 2**	**Tier 1**	**Tier 2**
AL.Algorithmic strategies	4	1		10	6			5	3
AL.Basic analysis			1		2				
AL.Basic automata, computability and complexity					2				1
AL.Fundamental data structures and algorithms		1		1	2	1			1
AR.Assembly level machine organization	4				2				1
AR.Interfacing and communication					3				1
AR.Machine level representation of data			1		1				
AR.Memory system organization and architecture			1		1				
DS.Basic logic	6			7				2	
DS.Basics of counting		1		9				3	
DS.Discrete probability				4	2			1	1
DS.Graphs and trees				10	2			5	1
DS.Proof techniques				9				4	
DS.Sets, relations, and functions				14				6	
HCI.Designing interaction	1				1				
IAS.Defensive programming	3			2		1		1	
IAS.Foundational concepts in security				2				1	
IAS.Principles of secure design							1		
IM.Information management concepts	1		1		3				2
NC.Introduction	6			2				1	
NC.Local area networks					5				2
NC.Networked applications		1		1					
NC.Reliable data delivery					2				1
NC.Resource allocation					3				2
NC.Routing and forwarding			3						
OS.Concurrency	5				1		1		1
OS.Memory management					1				
OS.Overview of operating systems						1			
OS.Scheduling and dispatch					2				2
OS.Security and protection					2				1

Table 6. *Cont.*

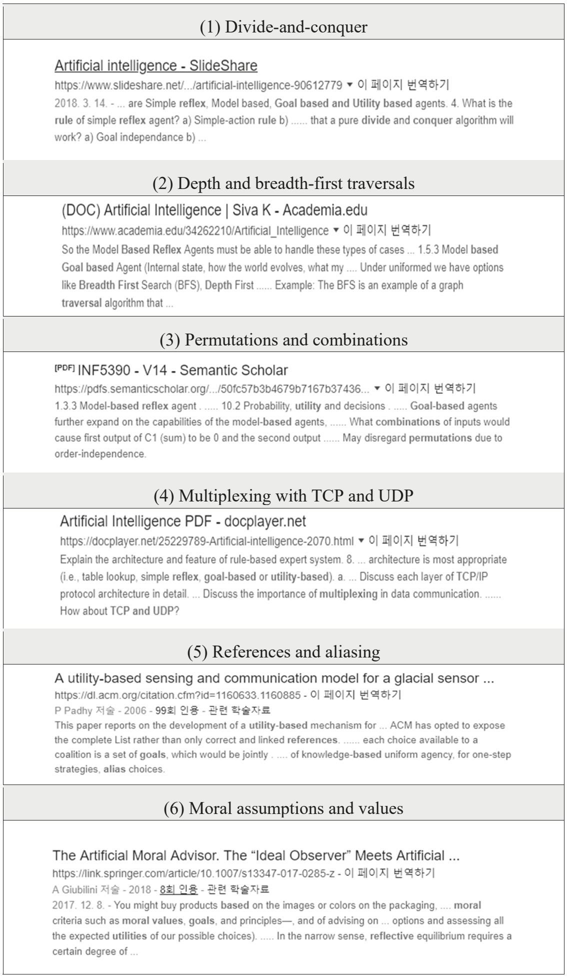
KA.KU (Knowledge Area. Knowledge Unit)		0.95 < Similarity				0.90 < Similarity ≤ 0.95			
		Fundamental Issues		Fundamental Issues		Basic Search Strategies		Basic Machine Learning	
		Tier 1	Tier 2	Tier 1	Tier 2	Tier 1	Tier 2	Tier 1	Tier 2
PD.Communication and coordination	4			2	5		1	1	2
PD.Parallel algorithms, analysis, and programming			1		2		1		1
PD.Parallel architecture				2	2			1	2
PD.Parallel decomposition					6		1		3
PL.Basic type systems	2			1					
PL.Functional programming				2				1	
SDF.Algorithms and design	3			6		1		3	
SDF.Development methods				5				2	
SDF.Fundamental data structures		1		9		1		4	
SE.Requirements engineering	6				3				2
SE.Software construction					2				1
SE.Software evolution			1		9		1		4
SE.Software processes				1					
SE.Software verification and validation					2				1
SE.Tools and environments					5				1
SF.Cross-layer communications	4			2		1		1	
SF.Parallelism				2					
SF.Reliability through redundancy									1
SF.State and state machines				2				1	
SP.Analytical tools	4	1		4				1	
SP.Intellectual property				2				1	
SP.Professional ethics			1		3				1
SP.Social context					2		1		1
Total	53	6	10	111	84	6	7	45	40

Table 7. Expert review results with regards to topics.

Range	Relevance	Necessity
0.95 < Similarity	3.74	3.74
0.95 ≥ Similarity > 0.9	3.49	3.58
0.9 ≥ Similarity > 0.8	3.08	3.20
0.8 ≥ Similarity > 0.7	3.07	3.14

4.4.2. Validation through Index Terms

Each of the six topics having semantic similarities in "reflexive, goal-based, and utility-based" and other areas from Tier 1 of "0.95 < similarity" was entered into a search engine and the results displayed on the first page were checked (see Figure 6). The search engine results are shown in Figure 8.



(1) Divide-and-conquer

Artificial intelligence - SlideShare
https://www.slideshare.net/.../artificial-intelligence-90612779 ▾ 이 페이지 번역하기
2018. 3. 14. - ... are Simple reflex, Model based, Goal based and Utility based agents. 4. What is the rule of simple reflex agent? a) Simple-action rule b) ...... that a pure divide and conquer algorithm will work? a) Goal independance b) ...

(2) Depth and breadth-first traversals

(DOC) Artificial Intelligence | Siva K - Academia.edu
https://www.academia.edu/34262210/Artificial_Intelligence ▾ 이 페이지 번역하기
So the Model Based Reflex Agents must be able to handle these types of cases ... 1.5.3 Model based Goal based Agent (Internal state, how the world evolves, what my .... Under uniformed we have options like Breadth First Search (BFS), Depth First ...... Example: The BFS is an example of a graph traversal algorithm that ...

(3) Permutations and combinations

[PDF] INF5390 - V14 - Semantic Scholar
https://pdfs.semanticscholar.org/.../50fc57b3b4679b7167b37436... ▾ 이 페이지 번역하기
1.3.3 Model-based reflex agent . ..... 10.2 Probability, utility and decisions . ..... Goal-based agents further expand on the capabilities of the model-based agents, ... What combinations of inputs would cause first output of C1 (sum) to be 0 and the second output ...... May disregard permutations due to order-independence.

(4) Multiplexing with TCP and UDP

Artificial Intelligence PDF - docplayer.net
https://docplayer.net/25229789-Artificial-intelligence-2070.html ▾ 이 페이지 번역하기
Explain the architecture and feature of rule-based expert system. 8. ... architecture is most appropriate (i.e., table lookup, simple reflex, goal-based or utility-based). a. ... Discuss each layer of TCP/IP protocol architecture in detail. ... Discuss the importance of multiplexing in data communication. ...... How about TCP and UDP?

(5) References and aliasing

A utility-based sensing and communication model for a glacial sensor ...
https://dl.acm.org/citation.cfm?id=1160633.1160885 - 이 페이지 번역하기
P Padhy 저술 - 2006 - 99회 인용 - 관련 학술자료
This paper reports on the development of a utility-based mechanism for ... ACM has opted to expose the complete List rather than only correct and linked references. ...... each choice available to a coalition is a set of goals, which would be jointly . .... of knowledge-based uniform agency, for one-step strategies, alias choices.

(6) Moral assumptions and values

The Artificial Moral Advisor. The "Ideal Observer" Meets Artificial ...
https://link.springer.com/article/10.1007/s13347-017-0285-z - 이 페이지 번역하기
A Giubilini 저술 - 2018 - 8회 인용 - 관련 학술자료
2017. 12. 8. - You might buy products based on the images or colors on the packaging, ... moral criteria such as moral values, goals, and principles—, and of advising on ... options and assessing all the expected utilities of our possible choices). ..... In the narrow sense, reflective equilibrium requires a certain degree of ...

Figure 8. Topic search results from search engine.

As shown in Figure 8, for (1), (2), (3), (4), and (6), words included in the two topics appeared together in the documents related to AI. This implies that the two topics are either AI-related or semantically close. Further, for the case of (5), "utility-based" and "alias, references," which are utility-based mechanisms for managing communications in collaborative multi-sensor networks, were found. From this, it can be interpreted that the terms appeared simultaneously in one document because communications using sensors are directly linked to the communications among agents.

5. Conclusions

The body of knowledge in a specific field of study reflects the linkages and continuity of knowledge. Identifying the inter-topic relationships among bodies of knowledge will help in constructing a hierarchy or a sequence of knowledge areas.

This study examined the interrelationships among the topics based on the understanding of various bodies of knowledge to provide suggestions for topic compositions to construct an academic body of AI knowledge. In constructing new topics through the analysis of the body of knowledge, sentence modeling, which is a data science method, was used to minimize noise in the semantic interpretation of the topics containing multiple words. Further, unprocessed original topics were applied. Regarding extracted contents, the validity was verified through expert validation tests and semantic similarity tests based on index terms.

Based on the obtained results, there were several topics with high similarity related to the agent design in the areas of "algorithm complexity," "discrete structures," "fundamentals of software development," "parallel and distributed computing," and "software engineering." These topics included algorithm design strategy, "divide-and-conquer," "data retrieval method," "depth and breadth-first traversals," and a theoretical foundation in the CS field, "permutations and combinations." The top three units with high similarity topic distribution based on KU were "sets, relations, and functions" and "graphs and trees" of DS, and "algorithmic strategies" of AL. These are the basic theories and concepts applied throughout the CS field, including the agent design.

Among the limitations of this study is sparsity of data on the core topics used in education. A large amount of data is needed to extract stable values through machine learning, but the ultimate limitation is that documents on the curriculum or core topics of education are not sufficient in terms of learning data. Furthermore, it is an area where research has not been conducted sufficiently on how to justify the results after the study through learning models. Consequently, identifying the possibilities through the educational use of the research results is time-consuming. To reflect emerging knowledge in education, the composition of knowledge or modeling of topics is essential. To achieve this, it is necessary to establish various methodologies to overcome limitations in the availability of extracted knowledge.

The field of AI deals with expert knowledge based on basic knowledge. The AI's body of knowledge compositions is extremely crucial in terms of preparing an academic foundation and deciding the topics to cover in the future. Therefore, the sentence modeling method presented in this study will contribute to the construction of various levels of knowledge hierarchies in the body of knowledge of AI and further understanding of the knowledge of the topics.

Author Contributions: Conceptualization, H.W. and J.K.; methodology, H.W.; software, H.W.; validation, H.W. and J.K.; writing—original draft preparation, H.W. and J.K.; writing—review and editing, H.W. and J.K.; visualization, H.W.; project administration, W.L.; funding acquisition, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2016R1A2B4014471).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Whitehouse Executive Order on AI. Available online: https://www.whitehouse.gov/ai/executive-order-ai/ (accessed on 2 March 2020).
2. Cabinet Office, Government of Japan Summit on Artificial Intelligence in Government. Artificial intelligence technology strategy meeting. Available online: https://www8.cao.go.jp/cstp/tyousakai/jinkochino/index.html (accessed on 2 March 2020).

3. *Ministry of Science and ICT I-Korea 4.0 Artificial Intelligence (AI) R&D Strategy for Realization*; Ministry of Science and ICT I: Korea, 2018.
4. *Carnegie Mellon University (CMU) B.S. in Artificial Intelligence*; Carnegie Mellon University (CMU): Pittsburgh, PA, USA, 2020.
5. MIT Artificial Intelligence: Implications for Business Strategy (self-paced online). Available online: https://executive.mit.edu/openenrollment/program/artificial-intelligence-implications-for-business-strategy-self-paced-online/ (accessed on 2 March 2020).
6. Ministry of Education, PRC Ministry of Education of People's Republic of China official Website. Available online: http://www.moe.gov.cn/srcsite/A16/s7062/201804/t20180410_332722.html (accessed on 2 March 2020).
7. State Council, PRC Notice of the State Council on Printing and Distributing a New Generation of AI Plan. Available online: http://www.gov.cn/zhengce/content/2017-07/20/content_5211996.html (accessed on 2 March 2020).
8. CBSE Central Board of Secondary Education. Available online: http://cbseacademic.nic.in/ai.html (accessed on 2 March 2020).
9. Cheng, J.; Huang, R.; Jin, Q.; Ma, J.; Pan, Y. An Undergraduate Curriculum Model for Intelligence Science and Technology. In Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China, 8–12 October 2018; pp. 234–239. [CrossRef]
10. Hearst, M. *Improving Instruction of Introductory AI*; Tech. Report FS-94-05; AAAI: Menlo Park, CA, USA, 1995.
11. Tungare, M.; Yu, X.; Cameron, W.; Teng, G.; Perez-Quinones, M.A.; Cassel, L.; Fan, W.; Fox, E.A. Towards a syllabus repository for computer science courses. In Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, KY, USA, 7–11 March 2007; pp. 55–59.
12. Dai, Y.; Asano, Y.; Yoshikawa, M. Course Content Analysis: An initiative step toward learning object recommendation systems for MOOC learners. In Proceedings of the 9th International Conference on Educational Data Mining, Raleigh, NC, USA, 29 June–2 July 2016; pp. 347–352.
13. Ida, M. Textual information and correspondence analysis in curriculum analysis. In Proceedings of the 2009 IEEE International Conference on Fuzzy Systems, Jeju Island, Korea, 20–24 August 2009; pp. 20–24.
14. Sekiya, T. Mapping analysis of CS2013 by supervised LDA and isomap. In Proceedings of the Teaching Assessment and Learning (TALE) 2014 International Conference, Wellington, New Zealand, 8–10 December 2014; pp. 33–40.
15. Matsuda, Y.; Sekiya, T.; Yamaguchi, K. Curriculum analysis of computer science departments by simplified, supervised LDA. *J. Inf. Process.* **2018**, *26*, 497–508. [CrossRef]
16. Jang, Y.; Kim, H. Reliable classification of FAQs with spelling errors using an encoder-decoder neural network in Korean. *Appl. Sci.* **2019**, *9*, 4758. [CrossRef]
17. Yin, X.; Zhang, W.; Zhu, W.; Liu, S.; Yao, T. Improving sentence representations via component focusing. *Appl. Sci.* **2020**, *10*, 958. [CrossRef]
18. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
19. Yoon, K. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1746–1751.
20. Shen, Y.; He, X.; Gao, J.; Deng, L.; Mesnil, G. Learning semantic representations using convolutional neural networks for web search. In Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, 7–11 April 2014.
21. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. *arXiv* **2014**, arXiv:1404.2188.
22. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
23. Mueller, J.; Thyagarajan, A. Siamese recurrent architectures for learning sentence similarity. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), Phoenix, AZ, USA, 12–17 February 2016; pp. 2789–2792.

24. Graves, A. Supervised Sequence Labelling. In *Studies in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2012.
25. Cho, K.; Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Schwenk, F.B.H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.
26. Socher, R. Recursive Deep Learning for Natural Language Processing and Computer Vision. Ph.D. Thesis, Stanford University: Stanford, CA, USA, 2014.
27. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1–15.
28. Langley, P.; Laird, J.E. *Artificial Intelligence and Intelligent Systems*; American Association for Artificial Intelligence: Menlo Park, CA, USA, 2013.
29. Aggarwal, C.C.; Hinneburg, A.; Keim, D.A. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001. ICDT 2001. Lecture Notes in Computer Science*; Bussche, J.V., Ed.; Springer: Berlin, Heidelberg, 2001; Volume 1973.
30. Pendlebury, J.; Humphrys, M.; Walshe, R. An experimental system for real-time interaction between humans and hybrid AI agents. In Proceedings of the 6th IEEE International Conference Intelligent Systems, Sofia, Bulgaria, 6–8 September 2012; pp. 121–129.
31. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson Education: London, UK, 2016.
32. Zaheer, K. *Artificial Intelligence Search Algorithms in Travel Planning*; Malardalen University: Västerås, Sweden, 2006.
33. Srinivasan, S.; Kumar, D.; Jaglan, V. Agents and their knowledge representations. *Ubiquitous Comput. Commun. J.* **2010**, *5*, 14–23.
34. Niederberger, C.; Gross, M.H. *Towards a Game Agent*; Institute of Visual Computing, ETH Zurich: Zurich, Sweden, 2002.
35. Ilsoon, S.; Buyeon, J.; JangHyung, J. Changes in E-Commerce and Economic Impact due to the Development of Agent Technology. *Inf. Commun. Policy Res. Rep.* 2001. Available online: https://www.nkis.re.kr:4445/subject_view1.do?otpId=KISDI00017852&otpSeq=0 (accessed on 2 March 2020).
36. Chowdhury, M. Emphasizing morals, values, ethics, and character education in science education and science teaching. *Malays. Online J. Educ. Sci.* **2016**, *4*, 1–16.
37. Poole, D.L.; Mackworth, A.K. *Artificial Intelligence Foundations of Computational Agents*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2017.
38. Cowan, C.; Wagle, P.; Pu, C.; Beattie, S.; Walpole, J. Buffer overflows: Attacks and defenses for the vulnerability of the decade. In Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, USA, 25–27 January 2000; pp. 119–129.
39. Banisar, D. *Whistleblowing: International Standards and Developments, Corruption and Transparency: Debating the Frontiers between State, Market. and Society*; World Bank-Institute for Social Research: Washington, DC, USA, 2011.
40. Cai, Z.; Wang, C.; Cheng, S.; Wang, H.; Gao, H. Wireless algorithms, systems, and applications. In Proceedings of the 9th International Conference, WASA 2014, Harbin, China, 23–25 June 2014.
41. Goralski, W. *The Illustrated Network How TCP/IP Works in a Modern Network*, 2nd ed.; Morgan Kaufmann: Burlington, MA, USA, 2017.
42. Bose, P.; Carufel, J.L.; Durocher, S.; Langerman, S.; Munro, I. Searching and routing in discrete and continuous domains. In *The Casa Matemática Oaxaca-BIRS Workshop*; Banff International Research Station: Banff, AB, Canada, 2015.
43. Lowry, M.R.; McCartney, R.D. *Automating Software Design*; MIT Press: Cambridge, MA, USA, 1991.

applied sciences

MDPI

Article

Prediction of Academic Performance at Undergraduate Graduation: Course Grades or Grade Point Average?

Ahmet Emin Tatar [1,†] and Dilek Düştegör [2,*,†]

[1] Department of Mathematics and Statistics, KFUPM, Dhahran 31261, Saudi Arabia; atatar@kfupm.edu.sa
[2] Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
* Correspondence: ddustegor@iau.edu.sa
† These authors contributed equally to this work.

Received: 22 June 2020; Accepted: 16 July 2020; Published: 19 July 2020

Abstract: Predicting the academic standing of a student at the graduation time can be very useful, for example, in helping institutions select among candidates, or in helping potentially weak students in overcoming educational challenges. Most studies use individual course grades to represent college performance, with a recent trend towards using grade point average (GPA) per semester. It is unknown however which of these representations can yield the best predictive power, due to the lack of a comparative study. To answer this question, a case study is conducted that generates two sets of classification models, using respectively individual course grades and GPAs. Comprehensive sets of experiments are conducted, spanning different student data, using several well-known machine learning algorithms, and trying various prediction window sizes. Results show that using course grades yields better accuracy if the prediction is done before the third term, whereas using GPAs achieves better accuracy otherwise. Most importantly, variance analysis on the experiment results reveals interesting insights easily generalizable: individual course grades with short prediction window induces noise, and using GPAs with long prediction window causes over-simplification. The demonstrated analytical approach can be applied to any dataset to determine when to use which college performance representation for enhanced prediction.

Keywords: academic performance; course grades; data mining; grade point average; machine learning; prediction; undergraduate

1. Introduction and Motivation

Educational Data Mining (EDM) is a fast-growing scientific field offering the potential to analyze a variety of student features to harness valuable knowledge from them. To this end, a plethora of predictive algorithms were effectively applied in educational contexts for numerous purposes using a variety of data and student records. As compiled in the review paper [1], two main application purposes can be identified in the college contexts: predictors and early warning systems (EWS). A predictor, "given a specific set of input data, aims to anticipate the outcome of a course or degree" [1], and a EWS "performs the same tasks as a predictor, and reports its findings to a teacher and/or to students at an early enough stage so that measures can be taken to avoid or mitigate potentially negative outcomes" [1]. Common prediction goals are listed as risk of failing a course, dropout risk, grade prediction, and graduation rate.

Among the various prediction goals, prediction of academic performance at graduation time especially, is of tremendous importance, as this information can be useful for:

- Enabling the educational institution to identify students (not) likely to complete the program and help in their admission decisions,
- Identifying students at risk and provide adequate advising and tailored help towards reduced failure rates,
- Detecting high achiever students and help them enhance their career paths,
- Analyzing factors of key importance and mobilize educational efforts for continuous quality improvements.

Looking at the literature on prediction of academic performance at the graduation time, we can observe that all studies rely mainly on four types of information on students, namely: (1) demographics and socio-economic, (2) high-school related, (3) college enrollment, and (4) college performance (up to the time of prediction).

Commonly used demographics and socio-economic information are sex/race [2], household income [3], age, first generation student [4], marital status, parents' jobs and educational levels [2]. Among the high-school related information, high-school GPA [2], pre-college marks [5,6], college admission test scores [3], public or private high-school [2], are frequently observed. As college related, in terms of enrollment information, the major and campus [2], a student's full-time vs. part-time status as well as whether s/he has a scholarship [3], enrolled hours and earned credit hours [4], year of entry and program [2,7] are often used. Finally, we observe that college performance has mostly been represented with grades from courses taken earlier [2,4,6,8], unless the prediction model is meant to be used at admission time [3,8,9].

Based on the above background, we observe that the bulk of previous studies used datasets with relatively large dimensionality of observations, some of them being expensive to measure (when not already available in the records). This, often combined with small samples, caused the curse of dimensionality, potentially yielding models with sub-optimal predictive power.

Recently, there is a trend to use only college performance [8,9], or using courses average per semester instead of individual courses grades (i.e., GPA per semester, or CGPA for cumulative average at time of prediction) [2,4,7]. However, there is no study comparing the performance of EDM models using individual course grades vs. grade point averages. It is unknown whether these two college performance representations are equivalent and can be used interchangeably, or if one is superior to the other in yielding better predictive power.

The main purpose of the present study, therefore, is to elucidate this matter by answering the following research question:

> *Is the individual course grade or grade point average more relevant for predicting student graduation academic performance?*

To answer this question, recent student data compiled at the College of Computer Science and Information Technology (CCSIT) from Imam Abdulrahman bin Faisal University (IAU) are used to generate two sets of predictive models, one using individual course grades, the other using the grade point averages. Thus, predictive power of respective models can be compared. However, it is well known that the performance of such models can also be affected by (1) student data used (besides the academic performance), (2) the data mining technique applied, as well as (3) how far from graduation the prediction is performed. Therefore, a comprehensive set of experiments is designed for spanning the whole search space made of student information besides the college performance, several machine learning methods commonly used in the literature, and prediction window of various sizes.

In the following sections, we first describe the research methodology, including, the dataset description, its preprocessing, the methods used, the experimental setup, and the evaluation criteria. Then, each conducted experiment and its results are reported, followed by the discussion and the concluding remarks.

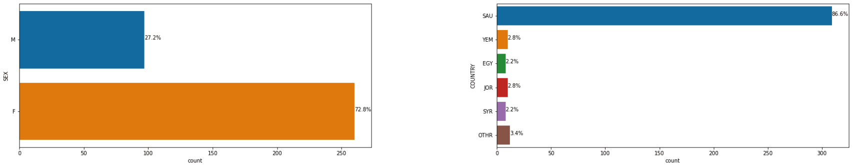
2. Research Methodology

2.1. The Dataset and its Preprocessing

Our dataset contains records of 357 students who were admitted to the CCSIT at IAU from Fall 2011 to Fall 2013 (included), and thus includes three batches of students. The institutional review board at IAU reviewed and approved using the data anonymously (application approved on 19 December 2018; IRB Number: 2018-09-304). Two programs of CCSIT are included in this study, namely Computer Science (CS) and Computer Information Systems (CIS). During the first three years, all the students of the College follow the same plan. In their first year, they attend the Preparatory program where they take mainly intensive English Language courses. In their second and third years, called General Years, the students take courses fundamental to computer and information sciences. At the end of their third year, students select either CS or CIS program based on their interests.

Student records populated from IAU learning management system contain features of three different nature: the demographic features, the pre-college features, and the college records including enrollment information and college performance.

The demographic features consist of gender and nationality (see Figure 1). The female gender dominates the CCSIT as the College is one of the top ranking colleges in the Eastern Region of Saudi Arabia for females. As expected, the dominant nationality is Saudi Arabian with over 85%. The other significant countries represented are Yemen (YEM), Egypt (EGY), Jordan (JOR), and Syria (SYR). There are nine other countries represented — Morocco, Pakistan, Palestine, Ethiopia, India, Iraq, USA, Bahrain, and Sudan —each with less than 0.5% grouped in the OTHR class.



(**a**) Gender Ratio (**b**) Country Ratio

Figure 1. Bar charts showing the demographic information of the dataset.

The demographic features being nominal, we need to convert them into numerical features to use them in machine learning models. We used three different approaches for this purpose.

1. The first approach is "*dummification*". For a nominal feature X with n categories, n new features $\{X_1, \ldots, X_n\}$, called *dummy* features, are created so that if a sample belongs to the i^{th} category, then $X_i = 1$ and $X_j = 0$ for all $1 \leq j \leq n$ and $i \neq j$.
2. The second approach is a derivative of the *dummification* approach. There is a redundancy between the new features $\{X_1, \ldots, X_n\}$. If we know the values of any $n - 1$ features, then we can find out the value the missing one. This redundancy may reduce the performance of a machine learning model. Therefore, in this variant of the dummification method, we drop one of the new features, the first one to be exact. We can summarize the dummification approach and its variant as substituting a nominal variable by a binary vector of size n and $n - 1$, respectively.
3. In case the dataset has too many nominal features with too many categories each and limited number of samples, the dummification of all the nominal features can reduce the performance of the machine learning algorithm as with the new features the number of total features can increase drastically. To avoid such complications, one can label a category of the nominal feature by its probability. This way, the total number of features do not change. For instance, in our dataset, this approach would have replaced the Female class by 0.728 and the Male class by 0.272.

We experimented with all three approaches. We did not see a major difference in the performance metrics when Logistic Regression or Random Forest machine learning methods are used. However we observed a significant decline in performance when Naive Bayes method is used with either the redundant or the non-redundant dummification which can be explained by the introduction of the new features that are not probabilistically independent. Because of this performance drop, we adopted the third approach in all our experiments.

The pre-college features consists of scores obtained from three national exams. These are numeric scores over 100. The only preprocessing we applied to these features were standardization. (i.e., shifted the mean to 0 and scaled the standard deviation to 1).

The academic records are the third group of features, which contain all transcript information, including admission term, graduation term, and letter grades for all the courses taken per term, for all terms including preparatory year until graduation. We only use the numerical values of the letter grades as described in Table 1. The irregular students, as they are very rare, are not included in this study. Thus, per semester, students take the courses as shown in Table 2 that is prepared based on the degree plan (The actual degree plans can be found at the links [10] for CS program and [11] for CIS program).

Table 1. Conversion table from ordinal to numerical value for letter grades (defined by the university).

Letter Grade	A+	A	B+	B	C+	C	D+	D	F
Numeric Grade	5	4.75	4.5	4	3.5	3	2.5	2	1

Table 2. Courses taken by term by regular CCSIT students during the academic years 2011/2012, 2012/2013, and 2013/2014.

Terms	**Course List**
1st Term	MATH 111, COMP 131, LRSK 141, PHEDU 162
2nd Term	ENGL 101, MATH 112, LRSK 142, COMP 122
3rd Term	CIS 211, CS 211, MATH 211, PHYS 212, ISLM 271
4th Term	CS 221, CS 222, STAT 207, BIOL 222, ISLM 272
5th Term	CIS 313, MATH 301, CS 311, CS 314, CIS 315, ISLM 273
6th Term	CS 310, CS 321, CIS 321, CIS 325, MGMT 290, CIS 413

The target variable in all the models is the graduation GPA, the weighted mean of the numeric scores of all the courses taken by a student. To draw more meaningful results, we use the graduation GPA not as a numerical feature but as an ordinal feature with three categories determined by the university. A student whose graduation GPA out of 5 is greater than or equal to 4.5 belongs to the class *"High GPA"*, between 4.5 and 3.75 (included) to the *"Average GPA"* class, and less than 3.75 to the *"Low GPA"* class. Figure 2 shows the distribution of three classes in the dataset.

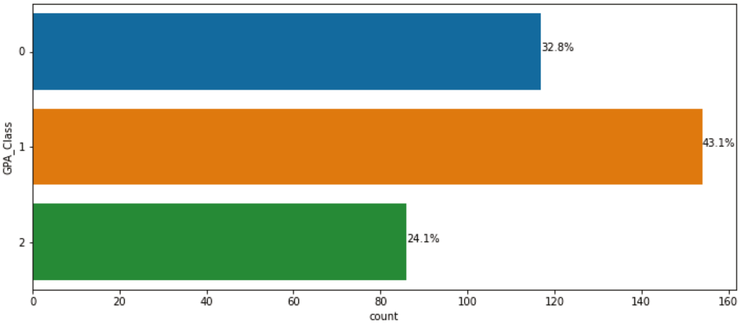


Figure 2. The bar chart showing the three classes in the target variable. The GPA Classes 0, 1, and 2 represent the classes Low GPA, Average GPA, and High GPA, respectively.

2.2. Experimental Set-Up

To answer the research question, we develop several classification models (and not regression, as the target variable has been transformed into an ordinal variable in Section 2.1) that differ with respect to (1) the way college performance is defined, (2) the type of student's data included, (3) the machine learning algorithm applied, (4) the size of the performance window, and (5) the size of the observation window (historical data).

In this study, we define the "term performance" of a student in two different ways. In the first representation, *by courses*, term performance is represented by a vector of size equal to the number of courses that should be taken in that term according to Table 2 with components being the numeric scores of the courses. In the second representation, *by GPA*, we represent term performance by the numeric weighted average of the courses with weights being the credit hours. Comparing the results obtained from the models *by courses* vs the models *by GPA* allows identifying which of the individual course grades and grade averages is more relevant for predicting student's graduation academic performance, thus answer the main research question of this study.

Then, the college performance data for students is modeled using two observation window size. In the first approach, only the immediate past term performance is included in the model (either last term *by courses*, or last term *by GPA*). In the second approach, a cumulative view is adopted where all the past terms' performance is included in the analysis (either cumulative *by courses*, or cumulative *by GPA*). The first approach corresponds to using only one term as history window, while the second approach corresponds to using all past terms data since the student joined the college. The reason to consider models that include last term performance only, is to isolate the term the most impactful to the student's success.

Figure 3 shows a sample student transcript data for the first 6 terms. For instance, let us consider predicting the graduation GPA class at the end of the second year, which is the end of the term 4, if we want to use one term observation history, then we use the term 4 performance alone, either as the term courses which is the vector $[2.5, 2, 4, 2.5, 4]$, or the GPA that is calculated as $(3 \times 2.5 + 4 \times 2 + 3 \times 4 + 4 \times 2.5 + 2 \times 4)/16 = 2.84375$. On the other hand, if we want to use all past observations cumulatively, then we use all past terms performance, either as all past courses, which is to say the vector $[4.5; 4.5; 4; 4; 4; 3.5; 4.75; 4.5; 3.5; 3; 2.5; 2.5; 4.75; 2.5; 2; 4; 2.5; 4; 2; 3; 2; 2.5; 3; 5; 2; 3; 2; 3.5; 3; 4.5]$, or the accumulated GPAs as $[4.3125; 4.1; 3.1; 2.84375]$.

For investigating the impact of the prediction window, we develop six models at different times of the curriculum, (1) as early as by the end of the first semester of the preparatory year, *term 1*, (2) by the end of preparatory year, *term 2*, (3) after the first semester of the general year, *term 3*, (4) by the end of the first general year, *term 4*, (5) after completing the first term of the second common year, *term 5*,

(6) by the end of the common general years, *term 6*. With reference to the student in Figure 3, the above described models correspond respectively to using only the term 1 data, adding one term at a time until all six terms data are used.

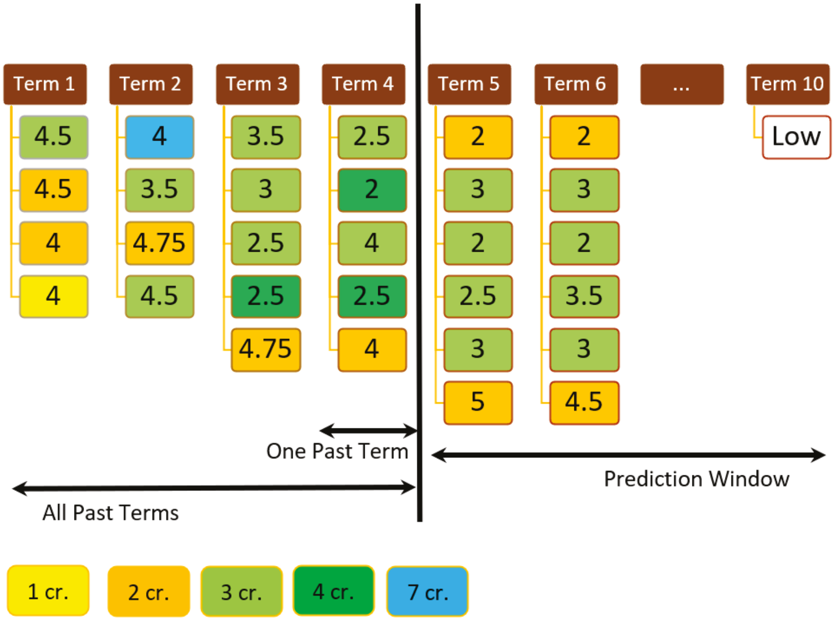


Figure 3. Sampe student transcript data (color code is, yellow: 1, orange: 2, light green: 3, dark green: 4, blue: 7 credits each).

We develop machine learning (ML) models working with the algorithm commonly used in EDM, namely Logistic Regression (LR), Random Forest (RF), and Naive Bayes (NB) with the accuracy as the performance metric. *Logistic Regression* is a linear model used for classification. It is often the first model considered due to its simplicity and interpretability. *Random Forest* is an ensemble method that fits number of decision trees. It makes a prediction based on the average of the predictions from the decision trees which is the method most used to predict graduation performance in the literature as identified in the review paper [12]. *Naive Bayes*, runner up method in the literature [12], is a statistical method based on the Bayes' Theorem. Its performance depends on the statistical independence of the features.

Finally, in order to investigate the impact of set of features on the model performance, we designed four experiments that exclude some features as seen in Table 3. Please note that we excluded all the four experiments which do not include academic records as they are not relevant to the goal of this study.

Table 3. Experiment scenarios (+ indicates inclusion of the feature set).

Terms	Demographics	Pre-College	Academic Records
Scenario 1 (S1)			+
Scenario 2 (S2)	+		+
Scenario 3 (S3)		+	+
Scenario 4 (S4)	+	+	+

Thus, a total of 288 experiments are conducted. Figure 4 recapitulates them.

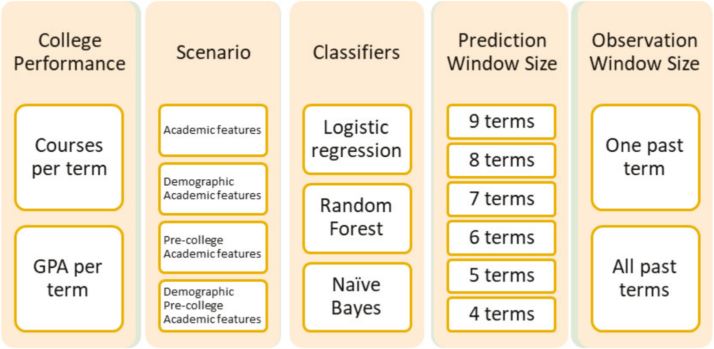


Figure 4. Experiments Conducted.

2.3. Performance Evaluation

As a base case model, instead of a simple random guess, we develop naive models based on the statistical facts that only uses term performance features and not any demographic or pre-college features. The idea behind these models is the following: for every term, we calculate the average term performance across all the students. No matter how these terms' performance is calculated, *Single-Course*, *Single-GPA* , *Cumulative-Course* , and *Cumulative-GPA*, if for a student they are always equal or above the mean of the term performance calculated across all the students, then that student is classified as *High GPA* student (i.e., a student always better than the average). Conversely, if they are always below, then the student is classified as *Low GPA* student (i.e., a student always lower than the average). All the other cases are classified as *Average GPA*. Table 4 illustrates the naive models with three sample students using the term performance by current GPA's.

Table 4. The table shows how Students A, B, and C are classified by the naive models by the end of Term 4, assuming the means of the first 4 term GPA's are 4, 3.25, 3.5, and 3.75, respectively.

	Term 1	**Term 2**	**Term 3**	**Term 4**	**Class (*Cumulative-GPA*)**	**Class (*Single-GPA*)**
Student A	4.1	3.5	3.5	4	High GPA	High GPA
Student B	3	3	3	3	Low GPA	Low GPA
Student C	4	3.2	3.75	4	Average GPA	High GPA

While developing the naive models, with a total of 357 samples, the size of the dataset can be problematic. If we use all the samples to develop our model, then we do not have any samples to estimate the true performance (performance of the model on an unseen data) of our models. Therefore, we split our dataset into training and testing datasets. We develop our model on the training dataset and evaluate its performance on the testing dataset. Performance indicators obtained using this approach, called *hold-out technique*, are more realistic. Yet, there are still some concerns. First of all, due to the hold-out samples, the learning is not 100%. To improve learning, we use split ratios with high training percentage. This creates yet another problem. Due to the small size of testing sets, the performance results may vary significantly. To reduce this variance, we can use repeated training and testing phases or use subsampling methods such as *k*-fold cross validation, or even repeated subsampling methods. We decide to use the repeated subsampling to minimize the variance in the accuracy scores. For this, the dataset is divided randomly into training and testing at the ratio of 4:1. Then, we calculate the statistics on the training dataset, do the classification of the samples on the testing dataset based on that statistics, and record the accuracy of the classification. Finally, we repeat this experiment 500 times and report the arithmetic mean as the result. Table 5 reports the performance of naives models.

Table 5. Results of the Naive Models.

		Term 1	Term 2	Term 3	Term 4	Term 5	Term 6
Single	**Course**	0.539	0.534	0.621	0.698	0.630	0.672
	GPA	0.460	0.498	0.556	0.553	0.561	0.558
Cumulative	**Course**	0.531	0.549	0.533	0.526	0.514	0.503
	GPA	0.470	0.618	0.671	0.736	0.734	0.742

3. Experiments and Results

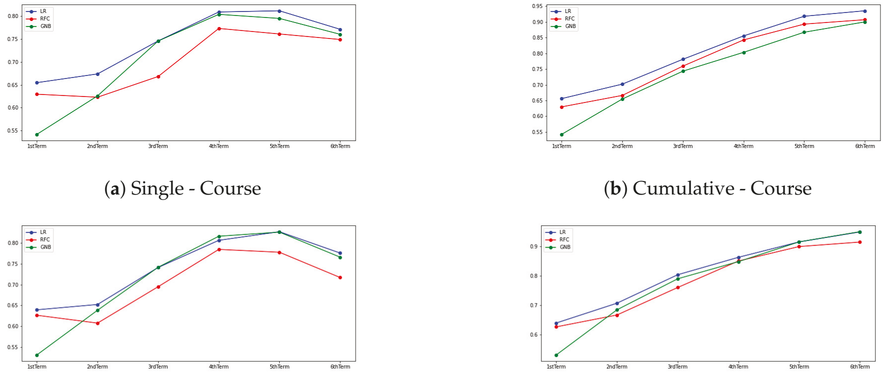
All experiments are done on Python 3. We design the ML models on Python's scikit-learn library version 0.22.2 with default hyper-parameters. For the ML models, we also use the repeated 5-fold stratified cross-validation with 100 repetitions. Since our goal is to observe the change of performance when the academic features are used either *by course* or *by GPA*, hyper-parameter search is not relevant. Nevertheless, when we tested random hyper-parameters, we observed the same trends as explained in the Discussion Section 4. We record the performance of the model both on the training and the testing datasets. All results are reported in following sub-sections per scenario.

3.1. Scenario 1: Academic Features Only

The first scenario only includes academic features, and the performance of the obtained prediction models are reported in Table 6. The best performance by the end of term 1, is the LR method with term performance represented *by courses*, whether single or cumulative, with 65.6%. When the prediction is performed later, the best performance is systematically obtained again with the LR method, but with the term performance represented *by cumulative GPA*. Please note that GNB shows the same best performance for the terms 5 and 6, and second best for the terms 3 and 4. Looking at Figure 5b,d, we observe that performance of the cumulative models are improving from 65.6% (term 1) to 94.9% (term 6) with decreasing prediction window size. Finally, Figure 5a,c show that among the models using only one past term, performance reaches a pick mostly when the prediction is performed by the end of term 4 or term 5.

Table 6. Accuracy results of the ML Models only with the academic features.

		ML Algoth	Term 1	Term 2	Term 3	Term 4	Term 5	Term 6
Single	**Course**	LR	**0.656**	0.673	0.746	0.809	0.812	0.774
		RFC	0.630	0.623	0.670	0.774	0.760	0.749
		GNB	0.542	0.625	0.745	0.804	0.796	0.760
	GPA	LR	0.639	0.652	0.741	0.8027	0.778	0.717
		RFC	0.626	0.607	0.695	0.785	0.778	0.717
		GNB	0.531	0.638	0.741	0.816	0.826	0.766
Cumulative	**Course**	LR	**0.656**	0.702	0.781	0.856	**0.918**	0.936
		RFC	0.630	0.666	0.760	0.843	0.893	0.907
		GNB	0.543	0.655	0.744	0.803	0.867	0.900
	GPA	LR	0.639	**0.707**	**0.803**	**0.863**	**0.915**	**0.949**
		RFC	0.626	0.667	0.760	0.850	0.899	0.915
		GNB	0.531	0.684	0.790	0.848	**0.915**	**0.949**



(**a**) Single - Course

(**b**) Cumulative - Course

(**c**) Single - GPA

(**d**) Cumulative - GPA

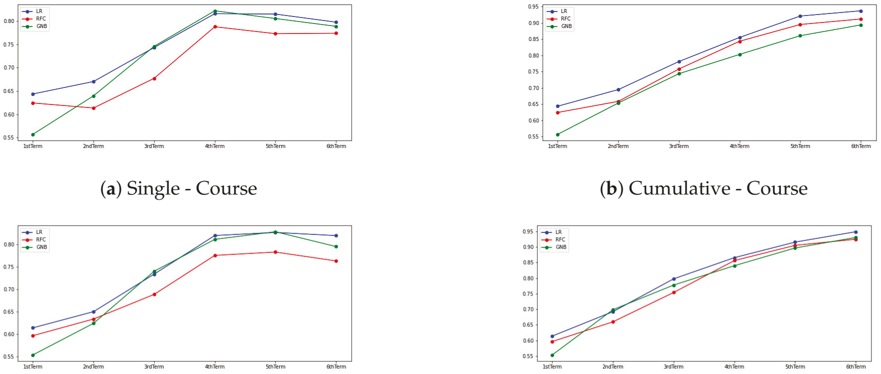
Figure 5. Accuracy plots along the terms of the ML Models only with the academic features.

3.2. Scenario 2: Demographics and Academic Features

The second scenario includes demographics and academic features. Performance of the obtained prediction models is reported in Table 7. The best performance by the end of term 1, is the LR method with the term performance represented *by courses*, whether single or cumulative, with 64.4%. When the prediction is performed later, we observe similar results with the scenario 1, i.e., the best performance is mainly obtained with the LR method, with the term performance represented *by cumulative GPA*. Please note that GNB shows a slightly superior performance for the term 4, and the second best for the terms 5 and 6. Looking at Figure 6b,d, we observe that the performance of cumulative models are improving from 64.4% (term 1) to 94.9% (term 6) with decreasing prediction window size. Again, same as for the scenario 1, the models using only one past term reach a performance pick mostly when the prediction is performed by the end of term 4 or term 5 (see Figure 6a,c).

Table 7. Accuracy results of the ML Models with the academic and demographic features.

		ML Algoth	1st Term	2nd Term	3rd Term	4th Term	5th Term	6th Term
Single	**Course**	LR	**0.644**	0.670	0.743	0.816	0.815	0.798
		RFC	0.624	0.614	0.677	0.788	0.773	0.774
		GNB	0.556	0.640	0.746	0.822	0.805	0.789
	GPA	LR	0.614	0.652	0.734	0.820	0.827	0.820
		RFC	0.597	0.634	0.689	0.776	0.783	0.764
		GNB	0.554	0.624	0.741	0.812	0.829	0.796
Cumulative	**Course**	LR	**0.644**	0.695	0.781	0.855	**0.921**	0.937
		RFC	0.624	0.659	0.758	0.843	0.895	0.912
		GNB	0.557	0.654	0.744	0.803	0.860	0.894
	GPA	LR	0.614	0.693	**0.798**	**0.866**	**0.916**	**0.949**
		RFC	0.597	0.660	0.754	0.857	0.905	0.925
		GNB	0.554	**0.699**	0.778	0.840	0.897	0.931



(**a**) Single - Course

(**b**) Cumulative - Course

(**c**) Single - GPA

(**d**) Cumulative - GPA

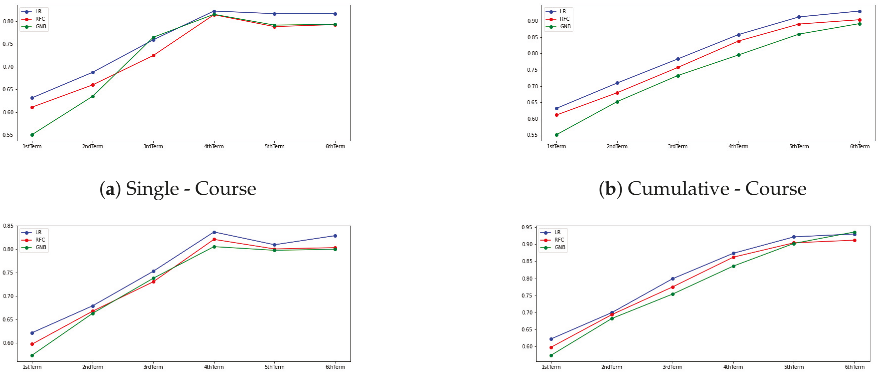
Figure 6. Accuracy plots along the terms of the ML Models with the academic and demographic features.

3.3. Scenario 3: Pre-College and Academic Features

The third scenario includes pre-college and academic features. Performance of the obtained prediction models is reported in Table 8. The best performance by the end of term 1, is again the LR method with term performance represented *by courses*, whether single or cumulative, with 63.2%. When the prediction is performed later, we observe similar results with the previous two scenarios, in other words the best performance is mainly obtained with the LR method, with term performance represented *by cumulative GPA*. The GNB shows a slightly superior performance for the term 6. Figure 7b,d shows that the performance of the cumulative models are improving from 63.2% (term 1) to 93.7% (term 6) with the decreased prediction window size. Again, same as for previous two scenarios, the models using only one past term reach a performance pick for prediction performed by the end of term 4 (see Figure 7a,c).

Table 8. Accuracy results of the the ML Models with the academic and pre-college features.

		ML Algoth	1st Term	2nd Term	3rd Term	4th Term	5th Term	6th Term
Single	**Course**	LR	**0.632**	0.688	0.759	0.822	0.817	0.817
		RFC	0.611	0.660	0.725	0.814	0.788	0.793
		GNB	0.551	0.635	0.765	0.815	0.791	0.793
	GPA	LR	0.622	0.679	0.753	0.837	0.810	0.829
		RFC	0.597	0.668	0.731	0.821	0.801	0.804
		GNB	0.574	0.663	0.738	0.806	0.798	0.800
Cumulative	**Course**	LR	**0.632**	**0.709**	0.783	0.858	0.912	0.931
		RFC	0.611	0.679	0.757	0.838	0.891	0.904
		GNB	0.551	0.652	0.732	0.795	0.860	0.892
	GPA	LR	0.622	0.699	**0.799**	**0.874**	**0.922**	0.931
		RFC	0.597	0.693	0.775	0.862	0.905	0.912
		GNB	0.574	0.682	0.754	0.836	0.903	**0.937**



(**a**) Single - Course

(**b**) Cumulative - Course

(**c**) Single - GPA

(**d**) Cumulative - GPA

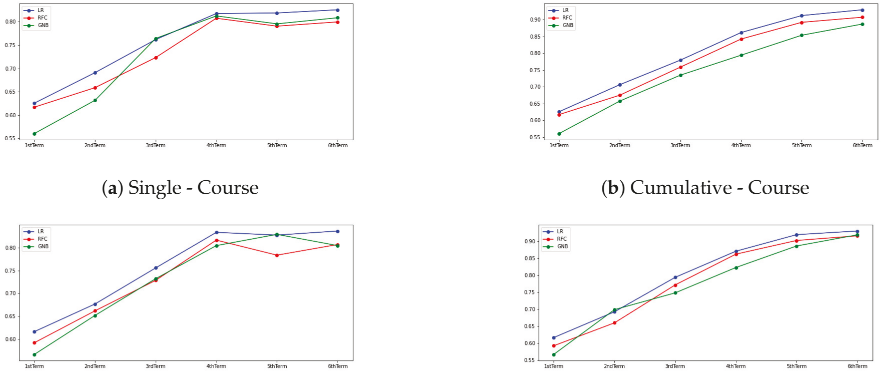
Figure 7. Accuracy plots along the terms of the ML Models with the academic and pre-college features.

3.4. Scenario 4: Demographics, Pre-College, and Academic Features

The last scenario includes all students data, that is to say, demographics, pre-college, and academic features. Performance of the obtained prediction models is reported in Table 9. The best performance by the end of term 1, is the LR method with the term performance represented *by courses*, whether single or cumulative, with 62.5%. When the prediction is performed later by the end of the terms 3, 4, 5, and 6, we observe similar results with all the past scenarios, that is, the best performance is mainly obtained with the LR method, with term the performance represented *by cumulative GPA*. Looking at Figure 8b,d, we observe that the performance of the cumulative models are improving from 62.5% (term 1) to 93.5% (term 6) with decreasing prediction window size. Models using only one past term reach a performance pick mostly when the prediction is performed by the end of term 4 or term 6 (see Figure 8a,c).

Table 9. Accuracy results of the ML Models with the academic, demographic, and pre-college features.

		ML Algoth	1st Term	2nd Term	3rd Term	4th Term	5th Term	6th Term
Single	**Course**	LR	**0.626**	0.691	0.762	0.818	0.819	0.826
		RFC	0.617	0.659	0.723	0.808	0.791	0.800
		GNB	0.560	0.632	0.764	0.813	0.796	0.809
	GPA	LR	0.616	0.679	0.757	0.830	0.821	0.836
		RFC	0.591	0.665	0.725	0.819	0.801	0.807
		GNB	0.565	0.651	0.732	0.804	0.798	0.802
Cumulative	**Course**	LR	**0.625**	**0.706**	0.779	0.861	0.912	**0.929**
		RFC	0.617	0.675	0.758	0.842	0.892	0.907
		GNB	0.560	0.657	0.735	0.794	0.853	0.887
	GPA	LR	0.616	0.697	**0.794**	**0.873**	**0.921**	**0.930**
		RFC	0.596	0.693	0.773	0.862	0.903	0.918
		GNB	0.565	0.687	0.751	0.825	0.887	0.919



(**a**) Single - Course

(**b**) Cumulative - Course

(**c**) Single - GPA

(**d**) Cumulative - GPA

Figure 8. Accuracy plots along the terms of the ML Models with the academic, demographic, and pre-college features.

4. Discussion

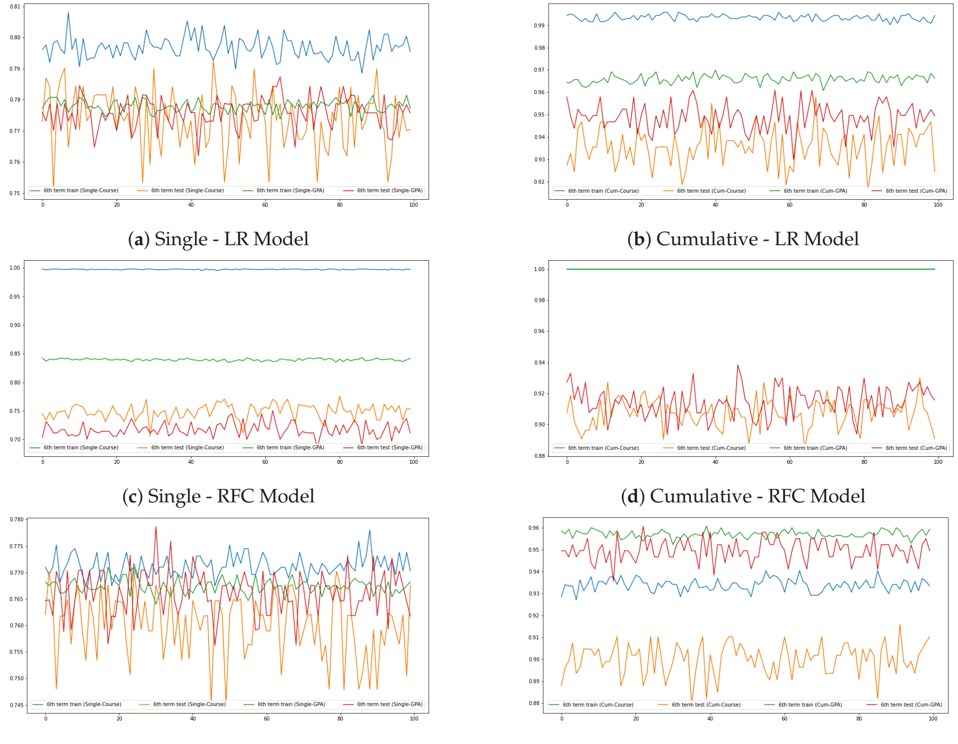
In this section, we are going to discuss our findings from two perspectives: (1) findings that can be generalized to any dataset and (2) findings specific to our dataset.

Looking at Table 10, which summarizes the best ML models per term, one can see the answer to the research question, namely, which representation of college academic performance is more relevant for predicting student graduation academic performance. When the models are compared based on how the academic performance is defined, that is whether *by courses* or *by GPA* as explained in Section 2.2, we notice, in general, that the models where the academic records are used *by GPA* achieve higher accuracy scores at the later terms whereas the models where the academic records are used *by courses* achieve higher accuracy scores at the earlier terms. To be more precise, we can see in Table 10 that course grades yields better results until the end of term 2, and GPA gives better results afterwards.

Table 10. Summary of the best models per term.

	Term 1	**Term 2**	**Term 3**	**Term 4**	**Term 5**	**Term 6**
Random Guess	0.333	0.333	0.333	0.333	0.333	0.333
Naive Model	0.539	0.618	0.671	0.736	0.734	0.742
Best cumulative ML Performance	0.656	0.709	0.803	0.874	0.922	0.949
Best cumulative ML Model	LR	LR	LR	LR	LR	LR
Academic performance	by course	by course	by GPA	by GPA	by GPA	by GPA
Scenario	S1	S3	S1	S4	S3	S1

However, in an attempt to gain more insight, we analyzed the variance of the training and testing performance of ML models. We observed that after term 2, the course models all had a higher variance. For example, Figure 9 illustrates the train and test accuracy scores by the number of experiments of all ML models on term 6. In all the plots, it is clearly visible that the difference between the green and the red curves (train and test of the GPA models) is less than the difference between the blue and the orange curves (train and test of the Course models). This indicates that if we introduce academic performance into the models *by courses*, in later terms, the models learn the noise (i.e., overfitting), which results in performance loss. This is not the case for GPA models in later terms, since one GPA per term would replace several (up to six at IAU-CCSIT) individual course grades, thus representing an equivalent information with reduced size of academic performance features. On the other hand, if we introduce the academic performance into the model as GPA at earlier terms, then we are over-simplifying the model by reducing the size of the academic features to a single number per term. We expect this observation to hold in any academic dataset and thus conclude that for earlier predictions academic performance should be used *by courses* and for later predictions *by GPA*.



(**a**) Single - LR Model

(**b**) Cumulative - LR Model

(**c**) Single - RFC Model

(**d**) Cumulative - RFC Model

(**e**) Single - GNB Model

(**f**) Cumulative - GNB Model

Figure 9. Variance plots of ML models for term 6.

The conducted experiments also allow answering several common EDM questions with respect to the specific IAU-CCSIT dataset. Our experiment results show that all the models perform better than a random guess (which can be considered to be roughly 33.3%). Moreover, ML models perform significantly better than the naive models that we defined in Section 2.3 as our base line result. We can conclude that the prediction models can be used as early as by the end of term 1, knowing that delaying the prediction thus gathering more academic data about the student will improve the performance of the classifier. Within ML models, LR shows the best overall performance with GNB being the runner up. As for the highest performance, both LR and GNB records 94.9% accuracy using all six current GPAs cumulatively. The poor performance of RFC can be explained by the overfitting which is very clear from Figure 9c,d where training performance reaches 100%. Looking at the four scenarios and the corresponding results, we observe that the demographics is not a significant group of features as scenario 2 never yields the best performance. Academic records alone give the best performance models when used by the end of term 1, term 3, and term 6. For the terms 2 and 5, knowing about pre-college exam results slightly improves the performance. Certainly, these results should be interpreted as specific to the IAU-CCSIT dataset and congruent with the many case studies in the field ([13] and in their references).

Finally, we can draw conclusions from the models where the academic records are used alone or cumulatively. As expected, when the academic records, again alone or with the other features, are used cumulatively we get the best accuracy scores. Yet, we can extract some valuable information from the models where the academic performance is used alone. For instance, from the accuracy graphs we see that the accuracy scores peaked at term 4 and stabilized afterwards. Hence, term 4, which is the end of the general year 1, has the maximum impact on the graduation GPA class. We can thus conclude that the term 4 is a good moment to start predicting the graduation performance. This information can also be shared with students explaining them that an extra effort they will put in their studies on term 4 will have higher impact on their graduation GPA.

5. Conclusions

With a plethora of studies in EDM for predicting a student's academic success at graduation time, this study investigated which of the individual course grades or grade averages is more relevant for predicting student graduation academic performance. Although both types of data are interchangeably used in the literature, there is no study comparing the performance of EDM models using grade averages vs. individual course grades. It is unknown when and how to use these two college performance representations to attain best predictive power. To elucidate this matter, a comprehensive set of experiments were conducted on the recent student data compiled from the second author's college.

The experiment results show that for earlier predictions, individual course grades should be used to represent academic performance, while it is preferable to use GPAs for prediction after a few terms. We explain based on variance analysis that this will help avoiding oversimplification and noise, as both can lower the performance of a predictive model. This is a novel contribution to the field of EDM, that will enable scientists and educators to decide which representation to adopt depending on the time of prediction.

The second main contribution of this study is to investigate the individual impact of each semester on the graduation academic performance. The results of such an analysis can help identifying when is the best time to do the prediction (e.g., in order not to miss the most impactful term), or can help in advising and motivating the students about when to put extra efforts in their studies.

Author Contributions: Conceptualization, D.D.; Methodology, A.E.T.; Software, A.E.T.; Validation, A.E.T.; Formal Analysis, A.E.T.; Investigation, D.D.; Resources, D.D.; Writing—Original Draft Preparation, A.E.T.; Writing—Reviewing—Editing, D.D.; Visualization, A.E.T.; Supervision, D.D.; Project Administration, D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research did not receive any grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liz-Domínguez, M.; Caeiro-Rodríguez, M.; Llamas-Nistal, M.; Mikic-Fonte, F.A. Systematic literature review of predictive analysis tools in higher education. *Appl. Sci.* **2019**, *9*, 5569. [CrossRef]
2. Miguéis, V.; Freitas, A.; Garcia, P.J.; Silva, A. Early segmentation of students according to their academic performance: A predictive modelling approach. *Decis. Support Syst.* **2018**, *115*, 36–51. [CrossRef]
3. Trussel, J.M.; Burke-Smalley, L. Demography and student success: Early warning tools to drive intervention. *J. Educ. Bus.* **2018**, *93*, 363–372. [CrossRef]
4. Mason, C.; Twomey, J.; Wright, D.; Whitman, L. Predicting Engineering Student Attrition Risk Using a Probabilistic Neural Network and Comparing Results with a Backpropagation Neural Network and Logistic Regression. *Res. High. Educ.* **2018**, *59*, 382–400. [CrossRef]
5. Asif, R.; Hina, S.; Haque, S.I. Predicting student academic performance using data mining methods. *Int. J. Comput. Sci. Netw. Secur.* **2017**, *17*, 187–191.
6. Asif, R.; Merceron, A.; Pathan, M. Predicting student academic performance at degree level: A case study. *Int. J. Intell. Syst. Appl.* **2014**, *7*, 49–61. [CrossRef]

7. Adekitan, A.I.; Salau, O. The impact of engineering students' performance in the first three years on their graduation result using educational data mining. *Heliyon* **2019**, *5*, e01250. [CrossRef] [PubMed]
8. Jiménez, F.; Paoletti, A.; Sánchez, G.; Sciavicco, G. Predicting the risk of academic dropout with temporal multi-objective optimization. *IEEE Trans. Learn. Technol.* **2019**, *12*, 225–236. [CrossRef]
9. Aluko, O.; Adenuga, O.; Kukoyi, P.; Aliu, S.; Oyedeji, J. Predicting the academic success of architecture students by pre-enrolment requirement: Using machine-learning techniques. *Constr. Econ. Build.* **2016**, *16*, 86. [CrossRef]
10. Computer Science Program. Available online: https://www.iau.edu.sa/en/colleges/college-of-computer-science-and-information-technology/programs/bachelor-of-science-in-computer-science-cs (accessed on 15 December 2019).
11. Computer Information Systems Program. Available online: https://www.iau.edu.sa/en/colleges/college-of-computer-science-and-information-technology/programs/bachelor-of-science-in-computer-information-systems-cis (accessed on 15 December 2019).
12. Alyahyan, E.; Düştegör, D. Predicting academic success in higher education: Literature review and best practices. *Int. J. Educ. Technol. High. Educ.* **2020**, *17*, 3. [CrossRef]
13. Alyahyan, E. Predicting undergraduate students' Success in a Saudi University Using Data Mining Techniques. Master's Thesis, Imam Abdulrahman bin Faisal University, Dammam, Saudi Arabia, 2019.

applied sciences

Article

Predicting and Interpreting Students' Grades in Distance Higher Education through a Semi-Regression Method

Stamatis Karlos, Georgios Kostopoulos and Sotiris Kotsiantis *

Department of Mathematics, University of Patras, 26504 Rio Patras, Greece; stkarlos@upatras.gr (S.K.); kostg@sch.gr (G.K.)

* Correspondence: sotos@math.upatras.gr

Received: 30 October 2020; Accepted: 24 November 2020; Published: 26 November 2020

Abstract: Multi-view learning is a machine learning app0roach aiming to exploit the knowledge retrieved from data, represented by multiple feature subsets known as views. Co-training is considered the most representative form of multi-view learning, a very effective semi-supervised classification algorithm for building highly accurate and robust predictive models. Even though it has been implemented in various scientific fields, it has not adequately used in educational data mining and learning analytics, since the hypothesis about the existence of two feature views cannot be easily implemented. Some notable studies have emerged recently dealing with semi-supervised classification tasks, such as student performance or student dropout prediction, while semi-supervised regression is uncharted territory. Therefore, the present study attempts to implement a semi-regression algorithm for predicting the grades of undergraduate students in the final exams of a one-year online course, which exploits three independent and naturally formed feature views, since they are derived from different sources. Moreover, we examine a well-established framework for interpreting the acquired results regarding their contribution to the final outcome per student/instance. To this purpose, a plethora of experiments is conducted based on data offered by the Hellenic Open University and representative machine learning algorithms. The experimental results demonstrate that the early prognosis of students at risk of failure can be accurately achieved compared to supervised models, even for a small amount of initially collected data from the first two semesters. The robustness of the applying semi-supervised regression scheme along with supervised learners and the investigation of features' reasoning could highly benefit the educational domain.

Keywords: educational data mining; student grade prediction; semi-regression; early prognosis; interpretation; COREG algorithm

1. Introduction

Educational data mining (EDM) has emerged in the past two decades as a highly-growing research field concerning the development and implementation of machine learning (ML) methods for analyzing datasets coming from various educational environments [1]. The key concept is to utilize these methods, extract meaningful knowledge about students' performance, and improve the learning process enriching the insights that the tutor may obtain on time. These methods are grouped into five main categories [2]: Prediction, clustering, relationship mining, discovery with models, and distillation of data for human judgment. The main research interest has been centered on predictive problems primarily concerned with three major questions [3]: (1) What outcome of students will be predicted? (2) Which ML methodology is the most effective for the specific problem? (3) How early can such a prediction be made?

Most of the EDM research is mainly focused on implementing supervised methods utilizing only labeled datasets. To this end, a plethora of classification and regression techniques have successfully been applied for predicting various learning outcomes of students, such as dropout, attrition, failure, academic performance, and grades, to name a few. In addition, the main interest concentrates on building efficient predictive models at the end of a course using all available information about students [4]. However, it is of practical importance to provide both accurate and early-step predictions at minimum cost [5]. A review of recent studies and developments in the field of EDM reveals that there is an urgent demand for accurate identification of students at risk of failure as soon as possible during the academic year, since early intervention activities and strategies can be implemented. Preventing academic failure, enhancing student performance, and improving learning outcomes is of utmost importance for higher education institutions that intend to provide high-quality education [6]. Some new directions that have recently been formatted concern the recognition of errors during the composition or the writing of code assessment, usually based on self-attenuation mechanisms for providing high quality automated debugging solutions to undergraduate and post-graduate students, as well as the exportation of remarkable insights about the obstacles that are met by them during such tasks [7].

Apart from supervised methods, semi-supervised learning (SSL) has gained a lot of attention among scientists in the past few years for solving a wide range of problems in various domains [8]. SSL methods exploit a small pool of labeled examples together with a large pool of unlabeled ones for building robust and highly-efficient learning models. However, SSL has not adequately used in the educational domain as easily identified after a thorough literature review. Nevertheless, some notable studies have emerged recently dealing with semi-supervised classification (SSC) tasks, such as student performance prediction or student dropout, while semi-supervised regression (SSR) is uncharted territory. The primal difference between SSC and SSR is that the target attribute is categorical in the former case, while a pure numeric quantity has to be predicted in the latter case. A recent literature review of SSR depicts the most important works in this field [9], separating them into approaches with a common strategy to solve their task, while more related works have been demonstrated on behalf of SSC [10].

Multi-view learning has also attracted the interest of this research community, distilling information from separate views, original or transformed ones, while a search of more appropriate subspaces into the initial feature set always remains a crucial learning task for boosting the performance of SSL methods [11,12]. Adopting ensemble learners has also been an active research territory concerning SSL [13], while some similar works have been demonstrated by our side [14,15]. Although some recent advances have taken place—exploiting graph-based solutions [16–18], or deep learning neural networks (DNNs) [19,20]—attempting to acquire more and more accurate predictions, or even robust ones in case that noisy inputs/labels have violated the ideal case of compact training data [21], such mechanisms introduce some important defeats:

- Increased computational issues regarding the size of the provided datasets;
- Operation under transductive mode with inefficient complexity for most real-life cases rejecting at the same time the extraction of an inductive mechanism as a generic solution;
- Inability to facilitate interpretability of the exported decisions/predictions [22,23].

The main scope of the present study is three-fold. At first, we implement a well-known semi-supervised regression algorithm that is based on multi-view learning, adopting several ML learners into its main kernel, tackling with the early prediction of undergraduate students' final exam grades in a one-year distance learning course. Each student is represented in terms of a plethora of features, which were collected from three different sources, thus producing three distinct sets of attributes: Demographics, academic achievements, and interaction within the course Learning Management System (LMS). Secondly, we investigate the effectiveness of the separate SSR variants that are produced compared with their corresponding supervised performance on the examined EDM task.

In this sense, the proposed model may serve as an early alert tool with a view to providing appropriate interventions and support actions to low performers.

Finally, we apply a well-established framework for acquiring trustworthy reasoning scores per included attribute/indicator into the original dataset. Hence, interpretable models are created, providing carefully computed explanations about the predicted grades ranking the importance of each indicator without any dimensionality reduction trick and avoiding overconsumption of computational resources under specific cases. To the best of our knowledge, this is the first completed study towards this direction [24], which hopefully will provide the basis for further research in the field of EDM, as it is stated in the relevant and conclusory Sections.

The remainder of this paper is organized as follows. In the next section, we discuss the need for explainable artificial intelligence (XAI) solutions to the field of EDM, highlighting some of the most important approaches in interpreting decisions/predictions of various learning models and the assets of the selected interpretability framework. Section 3 presents a brief overview of relevant studies in the EDM field and some recently published works related to the SSR task. The research goal is set in Section 4, together with an analysis of the dataset used in the experimental procedure. The total pipeline for applying a well-known COREG algorithm (CO-training REGressors) [25] as an SSR wrapper along with several ML learners and some DNNs variants is provided in Section 5, also describing the two distinct explaining mechanisms that are based on the computation of Shapley values [26]. The experimental process and results are presented in Section 6. Finally, our conclusions are drawn in Section 7, which also mentions some promising improvements to this seminal work.

2. Interpretability in Machine Learning

Consider the problem of predicting the final exam grade of students enrolled in a distance learning course using ML. In this case, a supervised algorithm is trained over a set of labeled data (the target attribute values are known), and an ML model is produced (supervised learning), which is subsequently deployed for predicting the grade of a previously unknown student for given values of the input attributes (features of students). The predictive model does not know why the student received the specific grade, while, at the same time, it fails to grasp the difference between anticipated grades and actual performance. Decision-makers are often hesitant to trust the results of these models, since their internal functions are primarily hidden in black-boxes [27]. This is quite reasonable, since people outside of the ML field neither can understand the manner that outputs are exported, nor are confident on just consuming some pure decisions without accompanying them with some consistent proofs. There is also a well-known trade-off regarding the predictive ability and the interpretability of ML algorithms, which unfortunately deters the co-existence of both these properties to be highly qualified under the same ML algorithm, in general. Since predictive models play a decisive role in the decision-making process in higher education institutions, the ability to comprehend these models seems to be indispensable. Thus, the interpretability of provided solutions usually needs to be filtered through XAI tools [28,29].

Model interpretability is the process of understanding the predictions of an ML model. In fact, it is the key point to build both accurate and reliable learning models. In traditional ML problems, the objective is to minimize the predictive error, while interpretability is focused on extracting more valuable information from the model [30]. Commonly, it aims to address questions, such as (Figure 1):

- What each attribute represents?
- Why was a specific prediction was made by the model?
- Which are the key factors of a specified prediction?
- Why a specific student was assigned a failing grade?
- Can we describe what the model has learned?
- How confident are we for the decisions of the model?

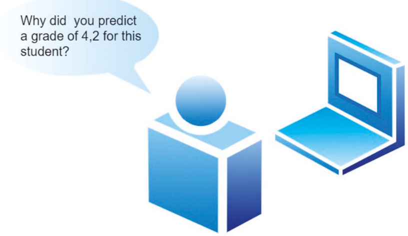


Figure 1. Why was a specific prediction was made by the model?

Although several published works have appeared in the literature of XAI recently, the majority of them make assumptions that are not actually consistent with the specifications of an educational task. For example, dimensionality reduction or feature transformations (e.g., semantic embeddings) may lead to incorrect conclusions or reasoning factors that ignore some of the underlying relationships that may be crucial for the real-life problem [31]. Furthermore, DNNs and their variants that operate by manipulating raw-data directly have highly attracted the interest of the XAI community, leading to solutions that are not applicable to our numerical source data. However, this fact does not exclude DNNs from being used as accurate black-boxes to such kind of problems, adopting mainly some model-agnostic approaches [32]. A representative work was done by Akusok et al. [22] exploiting extreme learning machines (ELM) trained on sampled subsets of the initial training set for increasing the output variance of the learning model, and later, explaining the information gained thought this strategy via proper confidence intervals for specific confidence levels. Both artificial and real-life datasets were evaluated, performing robust behavior without inducing much computational effort.

Besides DNNs, conventional ML algorithms need to overcome the long-standing obstacle of explainable predictions. One of the most popular libraries is LIME (local interpretable model-agnostic explanations) [33], which offers explanations based on local assumptions regarding the contribution of the examined learning model. A proper function that measures the interpretability and the local fidelity is defined, which is optimized using sparse linear models that are fed with perturbed samples from the region of interest. Global patterns are taken into consideration in the [32]. A framework of teacher-student models was proposed in Reference [34], where the corresponding explanations are obtained through adopting some additional models that mimic the behavior of the target black-box model and compare their performance on ground-truth trained models to clarify possible bias factors or reveal cases where the missing information has corrupted the final predictions. Because of the behavior of the adopted models, the confidence intervals are also produced for determining the importance of the detected differences.

Linear models and ensemble of trees were used in the previous work, while a solution that exploits some unsupervised mechanisms internally and focuses on exporting small, comprehensible, and more reliable rules exploiting ensemble of tress was proposed by Mollas et al. [35]. Both quantitative and qualitative investigation of the proposed LionForests approach has been taken place regarding Random Forest (RF) over binary classifications tasks, which is categorized as a local-based one. Another work that investigates classification tasks, but specializes in interpreting convolutional neural networks (CNNs) was recently demonstrated in Reference [36], where the Layer wise Relevance Propagation strategy was applied for extracting meaningful information when usual image transformations of audio signals are given as input. This process has been widely preferred for such networks, trying to propagate the computed weights of the total network to the input nodes, transforming them to important indications.

As it regards the adopted XAI framework by our side in the context of this work, Shapley values that stem from coalitional game theory constitute the basic concept that a more recent approach, named as Shapley additive explanations (SHAP), seems to satisfy better our research scope [37]. First, it is based on well-established theory and operates without violating a series of axioms: Efficiency, symmetry, dummy, and additivity. Without providing any extended analysis, we mention that Shapley values provide helpful insights by measuring the contribution of each feature into the original d-dimensional feature space $F \in \mathbb{R}^d$. Although this process demands quadratic computations regarding the size of F, it is an accurate and safe manner for revealing the actual contribution of each feature taking into consideration all the underlying dependencies of the measured values, thus assigning a combined profile of both local and global explanations. The exact formula for computing the total contribution of a random feature $i \in F$ through all the necessary weighted marginal contributions is given here:

$$contribution_i = \sum\nolimits_{S \subseteq F \setminus i} \frac{|S|!(d-|S|-1)!}{d!}(payout_{S \cup i} - payout_S) \tag{1}$$

$$payout_F = \int model(F) dF_{feature \notin F} - E_F(model(F)) \tag{2}$$

where each pay-out integrates the predictions of the selected model for any feature that belongs to the feature space F, while the rest ones are replaced by their mean value. In total, the Shapley values express the contribution that corresponds to each feature regarding the difference of the predicted value minus the average predicted value. Modifications that are more carefully implemented for obtaining the SHAP values reducing the overhead of the original procedure based on statistical assumptions or exploiting the nature of the base learner. Two such variants were adopted for facilitating the total efficacy of our methodology [26].

3. Related Work

Semi-regression has not been sufficiently implemented in the domain of EDM, as evidenced by a thorough study of the pertinent literature. Apparently, SSL classification algorithms cannot be directly applied for regression tasks, due to the nature of the target attribute, which is a real-valued one. Nevertheless, some recent and notable studies are discussed below.

Nunez et al. [38] proposed an SSR algorithm for predicting the exam marks of fourth-grade primary school students. The dataset comprised a wide range of students' information, such as demographics, social characteristics, and educational achievements. At first, the Tree-based Topology Oriented Self-Organizing Maps (TTOSOM) classifier was employed for building clusters exploiting all available data. These clusters were subsequently used for training the semi-regression model, which proved quite effective for handling the missing values directly without requiring a pre-processing stage. The experimental results demonstrated that the proposed algorithm achieved better results in terms of mean errors, compared to representative regression methods. Kostopoulos et al. [39] designed an SSR algorithm for predicting student grades in the final examination of a distance learning course. A plethora of demographic, academic, and activity attributes in the course Learning Management System (LMS) were employed, while several experiments were carried out. The results indicated the efficiency of the SSR algorithm compared to familiar regression methods, such as linear regression (LR), model trees (MTs), and random forests (RFs).

Bearing in mind the aforementioned studies and their findings, an attempt is made in the present study to implement an SSR algorithm for predicting the grades of undergraduate students in the final exams of a one-year online course offered by the Hellenic Open University. The main contribution of our research concentrates mainly on the following points:

- Semi-regression,
- Early prognosis, and
- Interpretation of features.

We also include some related works that concern the SSR field, which tackle problems from different domains. Besides the COREG algorithm [25], which inspired most of the upcoming SSR works on how to exploit unlabeled data for performing SSL methods for predicting numeric target attributes, the use of a co-training scheme did not found great acceptance for SSR works. We highlight just the direct expansion of COREG designed by Hady et al., via inserting the co-training by Committee for Regression (CoBCReg) scheme [40], which tries to encompass the use of more than one regressors for reducing noisy predictions, as well as the co-regularized least squares regression approach (CoRLSR) [41]. The latter one sets a risk minimization problem on the combined space of labeled and unlabeled data through proper kernel methods, focusing mainly on proposing some variants—a semi-parametric and a non-parametric—that scale linearly on the size of the unlabeled subset. The predictive benefits of adopting the co-training scheme without using any sophisticated feature split, just a random one, were remarkable.

More recently, a local linear regressor was employed by Liang R.-Z. et al. [42], which was iteratively applied for minimizing a joint problem on the neighborhood of each unlabeled examplFDe through sub-gradient descent algorithms. The authors of this work transformed two datasets that stem from unstructured data into structured problems and managed to outperform the compared algorithms regarding each posed performance metric, managing a competitive behavior regarding the time consumption. A multi-target fashion SSR model was presented in Reference [43], where the self-training scheme was combined with an efficient ensemble decision tree-based algorithm. Several modifications of the proposed scheme were examined, differentiated on the manipulation of the decisions that are drawn from the corresponding ensemble learner. Although their approach depends heavily on a reliability threshold which is domain-specific, a qualitative analysis was made over a dynamic selection, managing to outperform the supervised baseline as well as a random strategy for selecting unlabeled data for augmenting the initially collected data. Finally, an SSR method was used before applying an SSL method in the field of optical sensors, where limited data were readily available. In that scenario, a randomized method was used for generating unlabeled artificial data aiming at augmenting the labeled subset, but their annotation with pseudo-values was still crucial [44]. Therefore, a typical SSR strategy was applied before providing the finally created dataset to tackle the classification process.

4. Dataset Description

The dataset used in the research was provided by the Hellenic Open University and comprised records of 1073 students who attended the 'Introduction to Informatics' module of the 'Computer Science' course during the academic year 2013–2014.

These records were collected from three different sources, the course database, the teachers, and the course LMS, thus producing three distinct sets of attributes (Figure 2):

- The demographic set S_1 = {Gender, NewStudent} (Table 1).
  The distribution of male and female students was 76.5% and 23.5%, respectively. In addition, 87.5% of the students had enrolled in the course for the first time, while the rest failed to pass the previous year's final exams.
- The academic performance set $S_2 = \{\mathrm{Ocs}_i, \mathrm{Wri}_i\}_{i=1}^{2}$ (Table 2).
  The attribute named Ocs_i refers to students' absence or presence in the i-th optional contact session, while the attribute named Wri_i represents students' grades (ten-point grading scale) in the i-th written assignment, $i \in \{1, 2\}$. Four written assignments should be submitted during the academic year, while a total sum $\sum_{i=1}^{4} \mathrm{Wri}_i \geq 20$ was required for a student to undertake the final exam.
- The LMS activity set $S_3 = \{L_i, V_{1i}, V_{2i}, V_{3i}, V_{4i}, V_{5i}, P_{1i}, P_{2i}, P_{3i}, P_{4i}\}_{i=1}^{2}$ (Table 3).
  These attributes monitor student activity on the online LMS course (i.e., logins, views, and posts).

Table 1. Demographic attributes.

Attribute Name	Description	Values
Gender	Student gender	male, female
New Student	A student enrolled in the course for the first time	yes, no

Table 2. Academic performance attributes, i ∈ {1, 2}.

Attribute Name	Description	Values
Ocs_i	Student presence in the i-th optional contact session	absence, presence
Wri_i	Student grade in the i-th written assignment	[0, 10]

Table 3. LMS activity attributes in the i-th time-period, i ∈ {1, 2}.

Attribute Name	Description	Values
L_i	Total number of student logins	integer
V_{1i}	Number of student views in the pseudo-code forum	integer
V_{2i}	Number of student views in the compiler forum	integer
V_{3i}	Number of student views in the module forum	integer
V_{4i}	Number of student views in the course forum	integer
V_{5i}	Number of student views in the course news	integer
P_{1i}	Number of student posts in the pseudo-code forum	integer
P_{2i}	Number of student posts in the compiler forum	integer
P_{3i}	Number of student posts in the module forum	integer
P_{4i}	Number of student posts in the course forum	integer

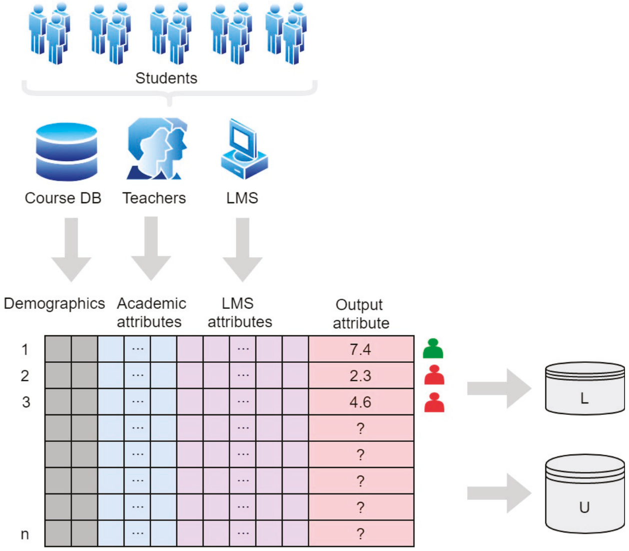


Figure 2. Gathering the data during the academic year.

Each instance of the dataset represents a single student (Figure 2) and is described by a vector of attributes, such as $x = (s_1, s_2, s_3)$, where s_1, s_2, s_3 correspond to the vector attributes of S_1, S_2, S_3 sets, respectively. Since the early prognosis of students at risk of failure is of utmost importance for higher education institutions, the academic year was divided into four time-periods according to each written assignment submission deadline (Figure 3). To this end, the notation V_{1i} denotes the total number of student views in the pseudo-code forum in the i-th period, i ∈ {1, 2}, and so forth. For example, attribute P_{21} refers to the total number of student posts in the compiler forum in the first time-period (i.e., from the beginning of the academic year until the first written assignment submission deadline). Finally, the output attribute $y \in [0, 10]$ represents the grade of students in the final examinations of the course. Note that we examine two distinct scenarios, corresponding to the first one and the first two time-periods, respectively.

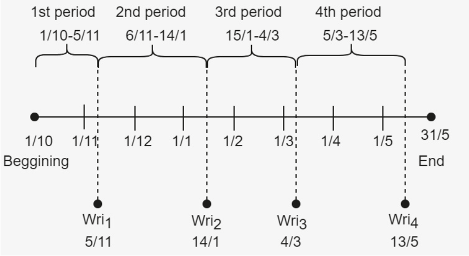


Figure 3. Time-periods of the academic year.

5. Proposed Semi-Supervised Regression Wrapper Scheme

Semi-Supervised Learning (SSL) is a rapidly evolving subfield of ML, embracing a wide range of high-performance algorithms. Typically, an ML model h is built from a training dataset $D = L \cup U$ consisting of a small pool of labeled examples $L = \{x_i,\ y_i\}_{i=1}^{l}$ and a large pool of unlabeled ones $U = \{x_i\}_{i=1}^{u}$, $l << u$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, $L \cap U = \varnothing$, without human intervention [45]. Depending upon the nature of the output attribute SSL is divided into two settings [9]:

- Semi-Supervised Classification (SSC).
  The labels y_i of the output attribute are discrete, i.e., $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$.
- Semi-Supervised Regression (SSR).
  The labels y_i of the output attribute are real, i.e., $\mathcal{Y} \subseteq \mathbb{R}$.

In our case, we employed an SSR scheme for exploiting the existence of both labeled and unlabeled data trying to acquire accurate estimations of the target attribute—students' final grade—based on a set of readily available data. Thus, one or more regressors are trained iteratively via selecting the most appropriate unlabeled data and annotating their missing target value in an automated fashion. Of course, the initial hypothesis is formatted on the manually gather the subset of L. Furthermore, the fact that the training set is split into two disjoint subsets, L and U, and that we aim at applying our trained model on another subset—the test set—which does not overlap with the training set leads us to an inductive SSR algorithm.

The most representative algorithm found in the literature that seems to satisfy our ambitions is the COREG that was firstly proposed by Zhou [25]. Actually, this learning scheme constitutes the analog of the co-training scheme also based on disagreement rule in the case of SSC [46], inserting a local-based criterion for measuring the effectiveness of the candidate unlabeled instances into the currently trained model for completing a regression task. Although various criteria have been designed in the context of SSC [47,48], the corresponding essential stage during an inductive SSR algorithm has not been highly studied by the related research community, following variants of the same criterion proposed in the case of COREG or proposing some new metrics that are mainly used under single-view works [44,49,50].

More specifically, the main concern of inductive SSR algorithms during the annotation of unlabeled examples is their *consistency* with the already existing labeled instances. This property is examined by measuring the next formula:

$$Consistency_{x_j} = \sum_{x_i \in L} \Big(f(y_i, h(x_i)) - f\big(y_i, h(\hat{x_i})\big) \Big), \ \forall x_j \in U \tag{3}$$

where f is a suitable performance metric, y_i is the actual value of the x_i labeled example, while $h(x_i)$ and $h(\hat{x_i})$ denote the output of regressor h when is trained solely on the current labeled set and on the augmented labeled set with the currently examined x_j example, respectively. According to the COREG algorithm, a local criterion is inserted for investigating if the consistency of each unlabeled example is beneficial for the current model per iteration. Thus, instead of examining the whole current L subset, only the neighbors of each $x_j \in U$ are considered for measuring the corresponding consistency metric, which is described in Equation (1). As it is discussed in the original work of the COREG, by maximizing this variant—mentioned hereinafter as $\delta_{x_j}\ \forall x_j \in U$—we reach safely either to the maximization of the general consistency metric or we acquire a zero value. In the first case, we pick the j-th unlabeled instance with a greater impact. Otherwise, we do not select any of them.

This strategy is similar to fitting an instance-based algorithm, like the k-Nearest Neighbors (kNN) [51], for selecting the unlabeled instances to augment the current labeled set per iteration, as it was preferred during the COREG approach. However, this fact does not hinder us from applying different kinds of regressors on the augmented labeled set, thus exploiting possible advantages of other learning models for capturing better the underlying relationships of the examined data. Based on our search in the literature, such a study has not yet been done.

Moreover, the already mentioned augmented per iteration labeled subset does not contain exclusively accurate values of the target attribute per its included instance, since during the training stage pseudo-labeled instances are joining the initially labeled examples, and their estimated values may differ from the actual one. This kind of noise into any SSL scheme may heavily deteriorate their total performance, settling them as myopic approaches that cannot guarantee safe predictions and violate the interpretation of the exported results.

Therefore, to alleviate the inherent confidence of COREG, we examine its efficacy on an EDM task that supports the multi-view description, increasing, thus the diversity of the trained regressors. Since the COREG algorithms is based on the co-training scheme, the feature space F of the original problem D is split into two disjoint views: $F = F_1 \cup F_2$. Although the random split has been proven quite competitive in several cases [52,53], co-training scheme should work if these two views are independent and sufficient.

The examined real-world problem brings a multidimensional and multi-view description that encourages us to train each regressor on separate views and get trustworthy predictions that would not harm our learning model regarding neither its predictiveness nor its interpretability despite the limited labeled data. Algorithm 1 presents the pseudocode of the end-to-end SSR pipeline.

Algorithm 1. The extended framework of the COREG algorithm.

```
Framework: Pool-based $COREG(D, selector1, selector2, regressor1, regressor2)$
Input:
  • Initially collected labeled $\boldsymbol{L} = \{x_i,\ y_i\}_{i=1}^{l}$ and unlabeled $\boldsymbol{U} = \{x_i\}_{i=1}^{u}$ instances, where $\boldsymbol{D} = \boldsymbol{L} \cup \boldsymbol{U}$ and $\boldsymbol{L} \cap \boldsymbol{U} = \varnothing$
  • $F_1, F_2$: provide the split of the original feature space F, where $F = F_1 \cup F_2$ and $F_1 \cap F_2 = \varnothing$
  • Define $Max\_iter$: maximum number of semi-supervised iterations and $f$: performance metric
Main process:
  • Set iter = 1, consistentSet = ∅
  • Train $selectori, regressori$ on $\boldsymbol{L}(F_i)\ \forall i \in \{1,2\}$
  • While iter ≤ Max_iter do
  •    For each $i \in \{1,2\}$ do
  •          For each $x_j \in \boldsymbol{U}$ do
  •             Compute $\delta_{x_j}(f)$ based on $selectori\ \forall i\ \in \{1,2\}$
  •             If $\delta_{x_j}(f) > 0$ : add j to consistentSet
  •       If consistentSet is empty do
  •             iter:= iter + 1 and continue to the next iteration
  •       else do
  •             Find the index $j^*$ of consistentSet s.t. $j^* = arg\ \max_{j} \delta_{x_j}$
  •             Update $\mathbf{U} :\ \mathbf{U}\ \leftarrow\ \mathbf{U} - \left\{x_{j^*}\right\}$
  •             For $i$ {1, 2} do
  •                Update $\mathbf{Li} :\ \mathbf{Li}\ \leftarrow\ \mathbf{Li} \cup \{x_{j^*},\ regressor\ \sim i\left(x_{j^*}\right)\}$, where ~i means the opposite index of the current
  •             Retrain $selectori, regressori$ on $\boldsymbol{L}(F_i) \forall i \in \{1,2\}$
  •             iter:= iter + 1
Output:
  • Apply the next rule to each met $x_{test}$ instance:
        $h_{COREG}(x_{test}) = 1/2 \cdot (regressor1(x_{test}) + regressor2(x_{test}))$
```

6. Experimental Process and Results

To conduct our experiments, we exploited the sci-kit Python library along with its integrated regressors and an implementation of computing the necessary Shapley values [37,54]. In order to systematically examine the efficiency of the extended COREG variant over the problem of early prognosis on student's performance, various choices of instance-based selectors and different learning model for the case of the regressors were chosen. Furthermore, we investigated two separate cases of the total dataset based on the measured indicators: Regarding only the first semester (D_1-first scenario) and only the first two semesters (D_2-second scenario). Thus, our predictions excuse the characterization of the early prognosis task, providing in time predictions using indicators that stem from the initial stages of an academic year. To be more specific, the size and the attributes of each view per dataset-scenario are reported here:

- First scenario:

$$D_1 = F_1 \cup F_2$$

$$|F_1| = 4,\ F_1 = (gender,\ NewStudent, Ocs_1, Wri_1)$$

$$|F_2| = 10, F_2 = (L_1, V_{11}, V_{21}, V_{31}, V_{41}, V_{51}, P_{11}, P_{21}, P_{31}, P_{41})$$

- Second scenario:

$$D_2 = F_1 \cup F_2$$

$$|F_1| = 6, F_1 = (gender,\ NewStudent, Ocs_1, Wri_1, Ocs_2, Wri_2)$$

$$|F_2| = 20, F_2 = (L_1, L_2, V_{11}, V_{12}, V_{21}, V_{22}, V_{31}, V_{32}, V_{41}, V_{42}, V_{51}, V_{52}, P_{11}, P_{12}, P_{21}, P_{22}, P_{31}, P_{32}, P_{41}, P_{42})$$

Besides the multi-view role of our extended COREG framework, the diversity of the SSR algorithm is enriched by the fact that each selector$_\mathrm{i}$ cannot select during one iteration the same x_{j^*} instance, while during the initial design of the COREG, randomly selected subsamples of the original U set were selected per iteration. Although we also attempted to implement this strategy, our results were constantly worse than the case of exploiting the full length of the original U set. This is probably due to the relatively small size of our total problem D, which we hope to undertake during the next semesters to enrich our collected data.

As it regards the choice of the investigated selectors and regressors for the extended COREG framework, we mention here all the different variants/models that were included in our experiments:

- (selector1, selector2): We have selected kNN algorithm for detecting the appropriate neighbors and fitting appropriate models. Following the original COREG scheme for injecting further diversity between the two separate views, we kept different power parameter for the internal distance that is exploited for formatting the neighborhood. Thus, we used Euclidean distance and Minkowski of 5th power for first and second selector, respectively. Moreover, we examined four separate cases based on the number of the nearest neighbors that are considered per case: (k_1, k_2) ≡ (1, 1), (1, 3), (3, 1), and (3, 3).
- (regressor1, regressor2): A different pair of same models have been used for this choice. To be more specific, we have used kNN with $k = 3$, a typical Linear regressor (LR), the Gradient Boosting regressor which is an additive model that operates under a forward-stage manner with 2 different loss functions: Least squares regression (ls) and 'huber'—a hybrid between ls and least absolute deviation—which are depicted as GB(ls) and GB(huber) and multi-layer perceptron that optimizes the squared-loss function by using the 'lbfgs' quasi-Newton solver. The last regressor is denoted as MLP, while its default hyperparameters were used: The 'Relu' as activation function and a hidden layer with 100 neurons. Although some further modifications of the internal parameters of each learner were investigated, as well as the combination of same learning models, but distinct regressors per view (e.g., train GB(ls) on $\boldsymbol{L}(F_1)$ and train GB(huber) on $\boldsymbol{L}(F_2)$), but neither this fact

serves our ambitions nor any great improvement was achieved. More information could be found in Reference [41].

As it concerns the rest required information about our evaluations, we set *Max_iter* equal to 100 and the performance metric f ≡ MSE (Mean Squared Error). Moreover, we applied a 5-fold-Cross-Validation (5-fold-CV) evaluation process, while we held 100 instances out of the 1073 as the test set. Consequently, the rest n = 973 instances constitute the D set, where the size of the L (l) and the U (u) subsets sum up to n. Thus, we examined four different values of the initially labeled instances: 50, 100, 150, and 200, while all of the rest instances were exploited from the first iteration as the U subset, since, as already mentioned, a possible random sampling of the total U subset per iteration did not favor us. Finally, the scenario under which our selectors exploit kNN algorithm with (k_1, k_2) = (1,1) did not manage to detect instances that satisfy the restriction of consistency as described in Equation (3) in the majority of the conducted experiments, and for this reason, was excluded by our results. The performance of the examined COREG variants based on the mean absolute error (MAE) metric is presented in Tables 4 and 5.

Table 4. Relative improvement of mean absolute error (MAE) metric (±std) of the dataset based only on the first semester during the best iteration per different combination of selector and regressor.

Size (L)	Selector$_i$			Regressor$_i$		
	(k_1, k_2)	3NN	LR	GB (ls)	GB (huber)	MLP
50	(1,3)	3.63 (±2.46)	10.06 (±6.42)	6.83 (±0.89)	3.78 (±1.62)	8.93 (±6.00)
	(3,1)	3.38 (±1.47)	10.51 (±8.12)	6.66 (±3.15)	5.07 (±1.09)	14.33 (±8.91)
	(3,3)	3.37 (±3.49)	15.27 (±8.15)	7.22 (±2.72)	4.60 (±2.19)	18.18 (±11.08)
100	(1,3)	2.28 (±3.00)	2.02 (±1.34)	1.81 (±1.66)	5.08 (±1.64)	5.82 (±4.49)
	(3,1)	2.33 (±1.70)	2.89 (±1.33)	1.95 (±1.63)	5.89 (±2.19)	6.14 (±5.11)
	(3,3)	2.26 (±2.76)	3.21 (±2.46)	2.42 (±2.84)	7.05 (±1.87)	9.42 (±4.77)
150	(1,3)	2.52 (±1.47)	0.63 (±0.72)	5.43 (±2.85)	3.32 (±2.14)	3.73 (±2.47)
	(3,1)	5.80 (±3/94)	1.07 (±1.62)	4.86 (±3.02)	3.44 (±1.28)	9.15 (±12.48)
	(3,3)	1.88 (±0.52)	2.07 (±1.53)	6.97 (±5.35)	2.67 (±1.31)	8.97 (±15.04)
200	(1,3)	2.83 (±1.82)	1.16 (±1.06)	3.31 (±2.53)	3.40 (±2.76)	4.95 (±2.46)
	(3,1)	4.04 (±3.41)	0.67 (±0.59)	2.93 (±2.56)	3.52 (±2.56)	3.05 (±1.94)
	(3,3)	0.88 (±1.05)	0.45 (±0.57)	4.58 (±3.34)	5.41 (±2.39)	5.85 (±2.66)

Table 5. Relative improvement of MAE metric (±std) of the dataset based only on the first and second semester during the best iteration per different combination of selector and regressor.

Size (L)	Selector$_i$			Regressor$_i$		
	(k_1, k_2)	3NN	LR	GB (ls)	GB (huber)	MLP
50	(1,3)	5.79 (±1.89)	21.85 (±8.10)	4.47 (±2.70)	7.55 (±2.12)	12.51 (±7.92)
	(3,1)	7.26 (±4.22)	22.65 (±7.89)	5.36 (±3.35)	7.70 (±3.99)	11.25 (±5.28)
	(3,3)	2.81 (±2.06)	30.69 (±13.83)	7.28 (±5.17)	8.40 (±3.72)	18.17 (±7.16)
100	(1,3)	5.56 (±1.85)	8.30 (±6.64)	6.00 (±1.18)	4.64 (±4.52)	9.26 (±7.55)
	(3,1)	3.65 (±2.72)	8.04 (±8.18)	6.62 (±2.61)	2.92 (±3.19)	6.91 (±2.8)
	(3,3)	1.91 (±2.16)	11.91 (±13.15)	7.57 (±2.50)	2.76 (±2.57)	10.77 (±8.35)
150	(1,3)	8.00 (±2.54)	6.50 (±2.71)	4.16 (±2.44)	6.64 (±3.00)	16.36 (±9.00)
	(3,1)	6.35 (±5.32)	6.03 (±2.90)	4.72 (±3.25)	6.47 (±3.87)	3.41 (±4.76)
	(3,3)	1.85 (±2.54)	9.46 (±6.59)	5.46 (±3.83)	8.57 (±5.82)	15.41 (±10.79)
200	(1,3)	2.35 (±2.53)	1.48 (±1.19)	3.94 (±2.37)	3.86 (±2.52)	13.25 (±6.64)
	(3,1)	1.80 (±1.02)	1.55 (±1.31)	3.88 (±2.76)	3.94 (±3.33)	7.21 (±6.20)
	(3,3)	1.64 (±1.31)	2.61 (±2.56)	5.91 (±5.13)	5.27 (±2.44)	15.36 (±10.65)

To be more specific, in these tables, we have recorded the relative improvement between the performance of each regressor during the initially provided labeled set, and the iteration that recorded the best performance until the criterion of either exceeding the *Max_iter* or not satisfying the consistency is violated. The results indicate that there is a decrease in the MAE metric, whilst the number of labeled instances is increased, as could be expected. Based only on the information regarding the first semester, it is noticed that the best performers are LR and MLP for size(L) = 50, while the tree-based learners achieved a more stable improvement over all the examined initially labeled subsets. Based on the information regarding both the first and the second semester, it is observed that the best performers are again LR and MLP for size(L) = 50, while they also performed greater improvement in the rest of the examined scenarios against their behavior on the previous case.

Additionally, we observe that as the cardinality of the L subset increases, the relative improvement of the investigated multi-view SSR approaches is decreasing in both cases during the majority of the recorded results. Through this kind of information, we can understand better the benefits of SSR approaches like COREG when multi-view problems are considered even under both limited labeled data are provided, and the volume of the unlabeled data is also highly restricted, reducing, thus the informativeness of this source of knowledge which is crucial for SSL scenario. Hence, the most important asset of transforming the COREG approach into a multi-view SSR variant is the remarkable reduction of the mean absolute error under strict conditions regarding the initially provided labeled instances. Despite the fact that the supervised learning performance in that cases is usually poor, since it heavily depends on the initially labeled data, both the insights that are obtained through the distinct, independent views and the disagreement mechanism that interchanges information between regressors that are fitted to these views lead to superior performance against it. Therefore, we believe that this indication is our most important contribution: Proof that in a real-life scenario, the complementary behavior of two separate views can be a trustworthy solution—even under highly limited labeled instances and not a large pool of unlabeled ones.

Another key is the fact that by mining additional unlabeled instances, we would expect even larger improvements in some cases, something that occurs by observing the fact that some approaches achieved their best performance at the late iterations, while almost none approach recorded its best performance during the early iterations. Thus, we are confident that by providing additional unlabeled instances, even better improvements should be achieved. Another interesting point that should be examined in the future is to insert a dynamic stage for terminating such a learning algorithm, avoiding saturation phenomena. A validation set could be useful, but small cardinality in a real-life dataset does not favor such a strategy.

Furthermore, in the majority of the presented results, we conclude that when the selectors coincide with the two 3NN algorithms, larger improvements of the relative error are recorded, especially for the more accurate models: GB-based variants and MLP. This happens due to the fact that in the majority of the cases that one selector coincides with the 1NN algorithm, this view through its fitted regressor does not detect any unlabeled instance that satisfies the consistency criterion. Hence, the other view is not actually enriched via the existence of annotated unlabeled instances. However, in the case of weaker regressors—3NN and LR—this behavior may be proven beneficial when noisy annotations take place, reducing, thus the chances of degeneration. To be complete with our experimental procedure, all our results are included in the following link: http://ml.math.upatras.gr/wp-content/uploads/2020/11/mdpi-Applied-Sciences-math-upatras-2020.7z, where the index of the best position per examined fold along with the improvement during the arbitrarily selected value of *Max_iter* are recorded per regressor based on the separate views, as well as the finally exported one. Furthermore, the supervised performance of the whole dataset D for both cases and each investigated regressor, as well as their performance on all the four separate initial versions of the L size, are included—facilitating each interested researcher about the efficacy of our approaches.

Regarding the interpretability of our results, we computed the Shapley values of each one of the five distinct regressors. To safely conclude that the COREG scheme can produce trustworthy explanations under the existence of limited labeled data per different learner, we made the next assumptions: We compared the purely supervised decisions of the total dataset evaluated with the aforementioned 5-fold-CV process per learner with the corresponding decisions that are exported by training the same regressor on the finally augmented L subset according to the adopted COREG scheme having fixed the choice of selector to (3NN, 3NN) with the pre-defined distance metrics as mentioned previously into this Section. Hopefully, in all the cases, we obtained similar enough decisions regarding the importance weights assigned to each indicator, while we had a perfect match between the ranking of the indicators. This fact verifies our main scope: To apply a multi-view SSR scheme that can improve the initial predictiveness of the model despite the limited number of the provided instances, acquiring at the same time trustworthy explanations about the importance of each included attribute.

Next, we present through suitable visualizations the SHAP values per case, exploiting the implementation provided by the authors of Reference [55]. Before we step to this stage, a short description is given regarding the two used approaches for computing these explainable weights that approximate the actual, but still computationally hungry Shapley values. First of all, a kernel-based approach was applied over all the five examined regressors (KernelSHAP), which is agnostic regarding the applied learning model and introduces a linear model that is fitted over the sampled pairs of (data, targets) and their generated weights. To generate these weights, several coalitions over the F space is produced, while the marginal distribution instead of the accurate conditional distribution is sampled for replacing the features that are absent during a random coalition. Although the assumptions here may lead to poor results because of the randomly selected coalitions that ignore some feature dependencies, the fact that a linear regression is applied during the last stage of the computation, additional strategies may easily be implemented trying to smoothen possible defects of this approximation (regularization, different learning model). On the other hand, a tree-based approach (TreeSHAP) has been applied in the case of GB-based approaches trying to figure out possible discrepancies between the explanation of this kind of learner. TreeSHAP constitutes an expansion of the KernelSHAP approximations, leading to faster results and facilitating the learners that are based on Decision Trees, integrating aggregating behavior through proper additive properties. Further information is provided in the original work [55].

We present here only the corresponding diagram of GB (ls) with both SHAP explainers, ignoring the similar enough performance of GB (huber), since it is the only tree-based regressor. The SHAP visualization plots (Figures 4–8) illustrate the attribute impact on the output of the produced regression model (the attributes are ranked in descending order from top to bottom) and how the attribute values impact the prediction (red color correlates to positive impact) in the first scenario using the D_1 dataset. Attributes Wri$_1$ (grade in the first written assignment), Ocs$_1$ (presence in the first optional contact session), and V$_{31}$ (number of views in the module forum) are the most important ones in all cases regardless of the regressor employed. In addition, these attributes seem to positively influence the target attribute (i.e., student grade in the final examinations). Therefore, high-achieving students in the first written assignment, students with high participation rates in the first optional contact session, and students with high view rates in the module forum achieve a higher grade in the final course exam. Very similar results were produced regarding the second scenario using the D_2 dataset. In this case, attribute Wri$_2$ (grade in the second written assignment) proved to be the most significant, along with attribute V$_{32}$ (number of views in the module forum).

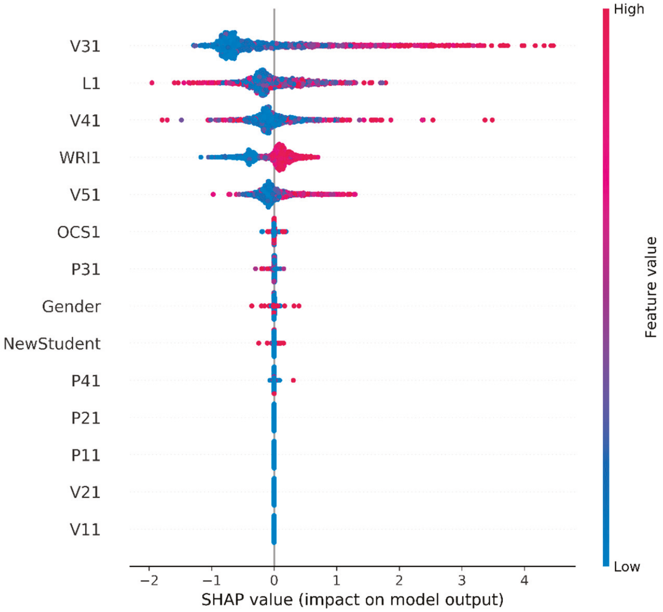


Figure 4. KernelSHAP values of the 5NN regressor (D_1 dataset).

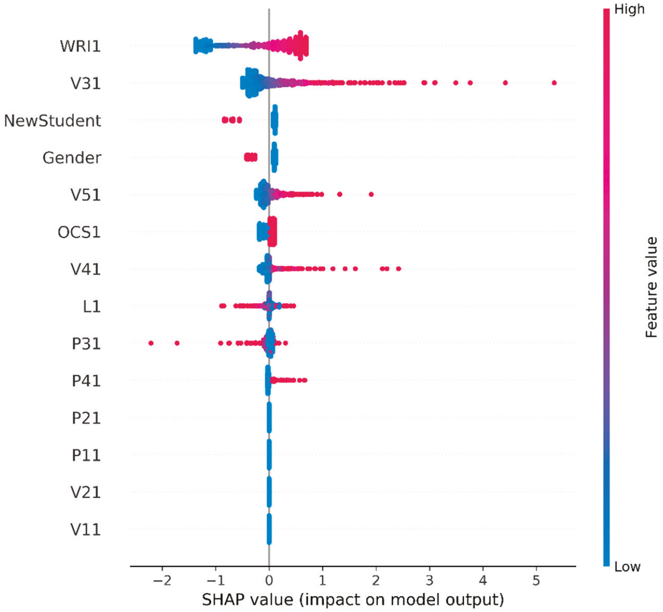


Figure 5. KernelSHAP values of the LR regressor (D_1 dataset).

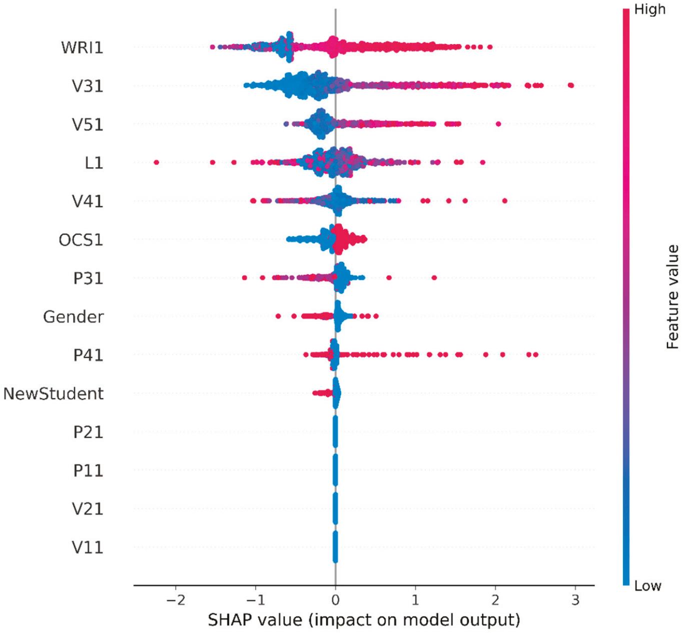


Figure 6. KernelSHAP values of the GB (ls) regressor (D_1 dataset).

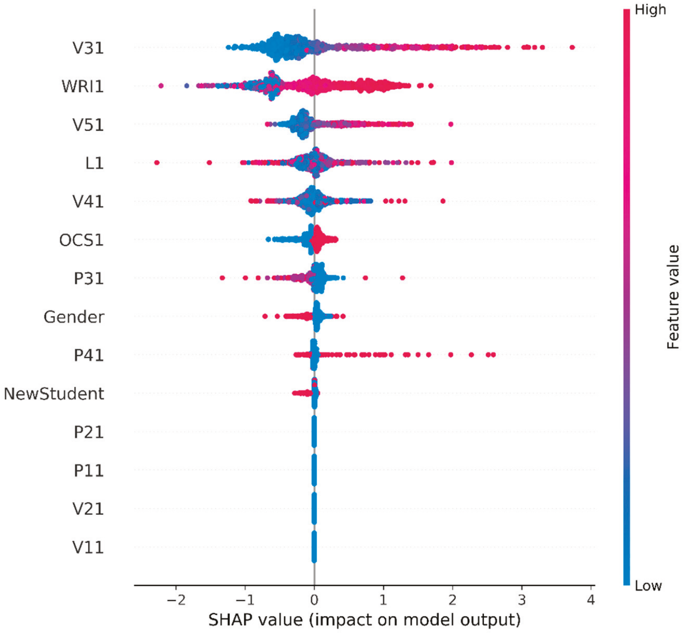


Figure 7. TreeSHAP values of the GB (ls) regressor (D_1 dataset).

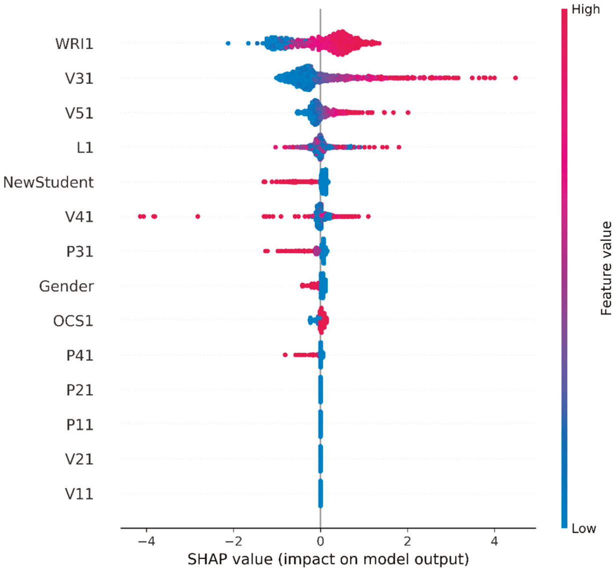


Figure 8. KernelSHAP values of the MLP regressor (D_1 dataset).

7. Conclusions

In the present study, an effort was made to build a highly-accurate semi-supervised regression model based on multi-view learning for the task of predicting student grades in a distance learning course. Additionally, we sought to gain insights and extract meaningful information from the model interpreting the predictions made and providing computed explanations about the predicted grades. The experimental results demonstrate the benefits brought by a natural split of the feature space. Therefore, our work contributes a different perspective to the existing single-view methods by fully exploiting the potential of different feature subsets by extending the COREG framework to the multi-view setting. In addition, it points out the importance of specific attributes that heavily influence the target attribute. Finally, the produced learning model may serve as an early alert tool for educators aiming at providing targeted interventions and support actions to low performers.

Generating synthetic data could be proven a highly favoring technique for mitigating the problem of limited labeled data. A recent demonstrated work has adopted such a strategy for training a boosting variant of the self-training scheme in the context of SSC [56]. In that work, the aspect of Natural Neighbors was preferred applying kNN algorithm as the base classifiers, and their obtained results seem encouraging enough for trying to extend their work also in our case. Another future direction could be applying pre-processing stages that may help us discriminate better the initially gathered data. Combination of semi-supervised Clustering either with conventional learners or ensembles, or even DNNs, as it has been validated in other real-life cases (e.g., geospatial data [57], medical image classification [58]) reducing inherent biases and helping us to uncover better possible underlying data relationships before the learning model could be found quite useful in practice. Another one possible effect of Clustering has been highlighted in Reference [50], where this strategy facilitated the scaling of a time-consuming learner over large volumes of unlabeled examples.

Finally, the strategy of transfer learning has been found great acceptance in the last years over several fields and could be proven beneficial in the case of EDM tasks. The two different aspects of this combination are expressed through either creating pre-trained models based on other learning tasks or enriching the discriminative ability of selected regressors through separate source domains that contain plentiful training data [59,60]. Combination of Active Learning with Semi-supervised learning might find great acceptance especially in cases that limited labeled data are provided, and the provided budget for monetization costs is highly bounded [61]. The modification also of transductive approaches for being considered under inductive learning scenarios seems a brilliant idea that compromises the accuracy of the former category and the generalization ability of the second one. Such a study was presented in Reference [62] and should be studied for SSR tasks.

Author Contributions: The authors contributed equally to the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baker, R.S.J.D.; Yacef, K. The state of educational data mining in 2009: A review and future visions. *JEDM J. Educ. Data Min.* **2009**, *1*, 3–17.
2. Baker, R. Data mining for education. *Int. Encycl. Educ.* **2010**, *7*, 112–118.
3. Costa, E.D.B.; Fonseca, B.; Santana, M.A.; De Araújo, F.F.; Rego, J.B.A. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Comput. Hum. Behav.* **2017**, *73*, 247–256. [CrossRef]
4. Márquez-Vera, C.; Cano, A.; Romero, C.; Noaman, A.Y.M.; Fardoun, H.M.; Ventura, S. Early dropout prediction using data mining: A case study with high school students. *Expert Syst.* **2016**, *33*, 107–124. [CrossRef]
5. Kostopoulos, G.; Karlos, S.; Kotsiantis, S.B. Multiview Learning for Early Prognosis of Academic Performance: A Case Study. *IEEE Trans. Learn. Technol.* **2019**, *12*, 212–224. [CrossRef]
6. Shelton, B.E.; Hung, J.-L.; Lowenthal, P.R. Predicting student success by modeling student interaction in asynchronous online courses. *Distance Educ.* **2017**, *38*, 59–69. [CrossRef]
7. Rahman, M.; Watanobe, Y.; Nakamura, K. Source Code Assessment and Classification Based on Estimated Error Probability Using Attentive LSTM Language Model and Its Application in Programming Education. *Appl. Sci.* **2020**, *10*, 2973. [CrossRef]
8. Zhu, X. *Semi-Supervised Learning Literature Survey*; University of Wisconsin-Madison: Madison, WI, USA, 2006.
9. Kostopoulos, G.; Karlos, S.; Kotsiantis, S.; Ragos, O. Semi-supervised regression: A recent review. *J. Intell. Fuzzy Syst.* **2018**, *35*, 1483–1500. [CrossRef]
10. Van Engelen, J.E.; Hoos, H. A survey on semi-supervised learning. *Mach. Learn.* **2019**, *109*, 373–440. [CrossRef]
11. Sun, S. A survey of multi-view machine learning. *Neural Comput. Appl.* **2013**, *23*, 2031–2038. [CrossRef]
12. Xu, C.; Tao, D.; Xu, C. A Survey on Multi-view Learning. *arXiv* **2013**, arXiv:1304.5634.
13. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Front. Comput. Sci.* **2020**, *14*, 241–258. [CrossRef]
14. Karlos, S.; Fazakis, N.; Kalleris, K.; Kanas, V.G.; Kotsiantis, S.B. An incremental self-trained ensemble algorithm. In Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Rhodes, Greece, 25–27 May 2018; pp. 1–8. [CrossRef]
15. Karlos, S.; Fazakis, N.; Kotsiantis, S.; Sgarbas, K. Self-Trained Stacking Model for Semi-Supervised Learning. *Int. J. Artif. Intell. Tools* **2017**, *26*. [CrossRef]
16. Fu, B.; Wang, Z.; Xu, G.; Cao, L. Multi-label learning based on iterative label propagation over graph. *Pattern Recognit. Lett.* **2014**, *42*, 85–90. [CrossRef]
17. Kang, Z.; Lu, X.; Yi, J.; Xu, Z. Self-weighted multiple kernel learning for graph-based clustering and semi-supervised classification. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 2312–2318. [CrossRef]
18. Wang, B.; Tsotsos, J.K. Dynamic label propagation for semi-supervised multi-class multi-label classification. *Pattern Recognit.* **2016**, *52*, 75–84. [CrossRef]
19. Luo, Y.; Ji, R.; Guan, T.; Yu, J.; Liu, P.; Yang, Y. Every node counts: Self-ensembling graph convolutional networks for semi-supervised learning. *Pattern Recognit.* **2020**, *106*, 107451. [CrossRef]

20. Ribeiro, F.D.S.; Calivá, F.; Swainson, M.; Gudmundsson, K.; Leontidis, G.; Kollias, S. Deep Bayesian Self-Training. *Neural Comput. Appl.* **2019**, *32*, 4275–4291. [CrossRef]
21. Iscen, A.; Tolias, G.; Avrithis, Y.; Chum, O. Label Propagation for Deep Semi-Supervised Learning. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 5065–5074.
22. Akusok, A.; Gritsenko, A.; Miche, Y.; Björk, K.-M.; Nian, R.; Lauren, P.; Lendasse, A. Adding reliability to ELM forecasts by confidence intervals. *Neurocomputing* **2017**, *219*, 232–241. [CrossRef]
23. Conati, C.; Porayska-Pomsta, K.; Mavrikis, M. AI in Education needs interpretable machine learning: Lessons from Open Learner Modelling. *arXiv* **2018**, arXiv:1807.00154.
24. Liz-Domínguez, M.; Caeiro-Rodríguez, M.; Llamas, M.; Mikic-Fonte, F.A. Systematic Literature Review of Predictive Analysis Tools in Higher Education. *Appl. Sci.* **2019**, *9*, 5569. [CrossRef]
25. Zhou, Z.-H.; Li, M. Semi-Supervised Regression with Co-Training. 2005. Available online: https://dl.acm.org/citation.cfm?id=1642439 (accessed on 31 October 2020).
26. Štrumbelj, E.; Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.* **2014**, *41*, 647–665. [CrossRef]
27. Wachter, S.; Mittelstadt, B.; Russell, C. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electron. J.* **2017**. [CrossRef]
28. Kanakaris, N.; Karacapilidis, N.; Kournetas, G. On the Exploitation of Textual Descriptions for a Better-informed Task Assignment Process. In Proceedings of the 9th International Conference on Operations Research and Enterprise Systems, {ICORES}, Valletta, Malta, 22–24 February 2020; Parlier, G.H., Liberatore, F., Demange, M., Eds.; SCITEPRESS: Setúbal, Portugal, 2020; pp. 304–310.
29. Chatzimparmpas, A.; Martins, R.M.; Jusufi, I.; Kerren, A. A survey of surveys on the use of visualization for interpreting machine learning models. *Inf. Vis.* **2020**, *19*, 207–233. [CrossRef]
30. Lipton, Z.C. The mythos of model interpretability. *Queue* **2018**, *16*, 31–57. [CrossRef]
31. Hosseini, B.; Hammer, B. Interpretable Discriminative Dimensionality Reduction and Feature Selection on the Manifold. *Lect. Notes Comput. Sci.* **2020**, *11906 LNAI*, 310–326. [CrossRef]
32. Plumb, G.; Molitor, D.; Talwalkar, A.S. Model Agnostic Supervised Local Explanations. *Adv. Neural Inf. Process. Syst.* **2018**, 2520–2529.
33. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should {I} Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 22nd {ACM} {SIGKDD} International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144. [CrossRef]
34. Tan, S.; Caruana, R.; Hooker, G.; Lou, Y. Distill-and-Compare: Auditing Black-Box Models Using Transparent Model Distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society—AIES '18*; ACM Press: New York, NY, USA, 2018; pp. 303–310. [CrossRef]
35. Mollas, I.; Bassiliades, N.; Vlahavas, I.P.; Tsoumakas, G. LionForests: Local interpretation of random forests. In *First International Workshop on New Foundations for Human-Centered AI (NeHuAI 2020)*; Saffioti, A., Serafini, L., Lukowicz, P., Eds.; CEUR: Aachen, Germany, 2020; pp. 17–24.
36. Houidi, S.; Fourer, D.; Auger, F. On the Use of Concentrated Time–Frequency Representations as Input to a Deep Convolutional Neural Network: Application to Non Intrusive Load Monitoring. *Entropy* **2020**, *22*, 911. [CrossRef]
37. Lundberg, S.M.; Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *Adv. Neural Inf. Process. Syst.* **2017**, 4768–4777.
38. Nuñez, H.; Maldonado, G.; Astudillo, C. Semi-supervised regression based on tree SOMs for predicting students performance. *IET Conf. Publ.* **2018**, *CP745*, 65–71. [CrossRef]
39. Kostopoulos, G.; Kotsiantis, S.; Fazakis, N.; Koutsonikos, G.; Pierrakeas, C. A Semi-Supervised Regression Algorithm for Grade Prediction of Students in Distance Learning Courses. *Int. J. Artif. Intell. Tools* **2019**, *28*, 1940001. [CrossRef]
40. Hady, M.; Schwenker, F. Co-Training by Committee: A Generalized Framework for Semi-Supervised Learning with Committees. *Int. J. Softw. Inform.* **2008**, *2*, 95–124. [CrossRef]
41. Brefeld, U.; Gärtner, T.; Scheffer, T.; Wrobel, S. Efficient co-regularised least squares regression. In Proceedings of the 23rd International Conference on World Wide Web-WWW '14, Seoul, Korea, 7–11 April 2006; pp. 137–144.

42. Liang, R.Z.; Xie, W.; Li, W.; Du, X.; Wang, J.J.Y.; Wang, J. Semi-supervised structured output prediction by local linear regression and sub-gradient descent. *arXiv* **2016**, arXiv:1606.02279.
43. Levatić, J.; Ceci, M.; Kocev, D.; Džeroski, S. Self-training for multi-target regression with tree ensembles. *Knowledge-Based Syst.* **2017**, *123*, 41–60. [CrossRef]
44. Kim, S.W.; Lee, Y.G.; Tama, B.A.; Lee, S. Reliability-Enhanced Camera Lens Module Classification Using Semi-Supervised Regression Method. *Appl. Sci.* **2020**, *10*, 3832. [CrossRef]
45. Chapelle, O.; Scholkopf, B.; Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Trans. Neural Networks* **2009**, *20*, 542. [CrossRef]
46. Zhou, Z.-H.; Li, M. Semi-supervised learning by disagreement. *Knowl. Inf. Syst.* **2010**, *24*, 415–439. [CrossRef]
47. Barreto, C.A.S.; Gorgônio, A.; Canuto, A.M.P.; João, C.X., Jr. A Distance-Weighted Selection of Unlabelled Instances for Self-training and Co-training Semi-supervised Methods. In *BRACIS*; Springer: Cham, Switzerland, 2020; pp. 352–366. [CrossRef]
48. Liu, Y.; Wang, L.; Mammadov, M. Learning semi-lazy Bayesian network classifier under the c.i.i.d assumption. *Knowledge-Based Syst.* **2020**, *208*, 106422. [CrossRef]
49. Fazakis, N.; Karlos, S.; Kotsiantis, S.; Sgarbas, K. A multi-scheme semi-supervised regression approach. *Pattern Recognit. Lett.* **2019**, *125*, 758–765. [CrossRef]
50. Guo, X.; Uehara, K. Graph-based Semi-Supervised Regression and Its Extensions. *Int. J. Adv. Comput. Sci. Appl.* **2015**, *6*. [CrossRef]
51. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Wang, R. Efficient kNN Classification with Different Numbers of Nearest Neighbors. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 1774–1785. [CrossRef]
52. Karlos, S.; Kanas, V.G.; Aridas, C.; Fazakis, N.; Kotsiantis, S. Combining Active Learning with Self-train algorithm for classification of multimodal problems. In Proceedings of the 10th International Conference on Information, Intelligence, Systems and Applications (IISA), Patras, Greece, 15–17 July 2019; pp. 1–8. [CrossRef]
53. Nigam, K.; Ghani, R. Understanding the Behavior of Co-training. *Softwarepract. Exp.* **2006**, *36*, 835–844.
54. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Vanderplas, J. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2012**, *12*, 2825–2830. [CrossRef]
55. Lundberg, S.M.; Erion, G.; Chen, H.; Degrave, A.; Prutkin, J.M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S.-I. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2020**, *2*, 56–67. [CrossRef] [PubMed]
56. Li, J.; Zhu, Q. A boosting Self-Training Framework based on Instance Generation with Natural Neighbors for K Nearest Neighbor. *Appl. Intell.* **2020**, *50*, 3535–3553. [CrossRef]
57. Yao, J.; Qin, S.; Qiao, S.; Che, W.; Chen, Y.; Su, G.; Miao, Q. Assessment of Landslide Susceptibility Combining Deep Learning with Semi-Supervised Learning in Jiaohe County, Jilin Province, China. *Appl. Sci.* **2020**, *10*, 5640. [CrossRef]
58. Peikari, M.; Salama, S.; Nofech-Mozes, S.; Martel, A.L. A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification. *Sci. Rep.* **2018**, *8*, 1–13. [CrossRef]
59. Tsiakmaki, M.; Kostopoulos, G.; Kotsiantis, S.B.; Ragos, O. Transfer Learning from Deep Neural Networks for Predicting Student Performance. *Appl. Sci.* **2020**, *10*, 2145. [CrossRef]
60. Wang, G.; Zhang, G.; Choi, K.-S.; Lam, K.-M.; Lu, J. Output based transfer learning with least squares support vector machine and its application in bladder cancer prognosis. *Neurocomputing* **2020**, *387*, 279–292. [CrossRef]
61. Karlos, S.; Kostopoulos, G.; Kotsiantis, S.B. A Soft-Voting Ensemble Based Co-Training Scheme Using Static Selection for Binary Classification Problems. *Algorithms* **2020**, *13*, 26. [CrossRef]
62. Yi, Y.; Chen, Y.; Dai, J.; Gui, X.; Chen, C.; Lei, G.; Wang, W. Semi-Supervised Ridge Regression with Adaptive Graph-Based Label Propagation. *Appl. Sci.* **2020**, *8*, 2636. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Social Media Rumor Refuter Feature Analysis and Crowd Identification Based on XGBoost and NLP

Zongmin Li [1], Qi Zhang [1], Yuhong Wang [2] and Shihang Wang [1,*]

[1] Business School, Sichuan University, Chengdu 610065, China; lizongmin@scu.edu.cn (Z.L.); 2019225020006@stu.scu.edu.cn (Q.Z.)

[2] USC-SJTU Institute of Cultural and Creative Industry, Shanghai Jiaotong University, Shanghai 200240, China; honglexi@sjtu.edu.cn

* Correspondence: sw3275@columbia.edu; Tel.: +86-028-8541-5143

Received: 12 May 2020; Accepted: 6 July 2020; Published: 8 July 2020

Featured Application: Results of this work can be applied to anti-rumor microblog recommendation decisions for social media platforms, in order to reduce the impact of rumors by promoting the spread of the truth.

Abstract: One prominent dark side of online information behavior is the spreading of rumors. The feature analysis and crowd identification of social media rumor refuters based on machine learning methods can shed light on the rumor refutation process. This paper analyzed the association between user features and rumor refuting behavior in five main rumor categories: economics, society, disaster, politics, and military. Natural language processing (NLP) techniques are applied to quantify the user's sentiment tendency and recent interests. Then, those results were combined with other personalized features to train an XGBoost classification model, and potential refuters can be identified. Information from 58,807 Sina Weibo users (including their 646,877 microblogs) for the five anti-rumor microblog categories was collected for model training and feature analysis. The results revealed that there were significant differences between rumor stiflers and refuters, as well as between refuters for different categories. Refuters tended to be more active on social media and a large proportion of them gathered in more developed regions. Tweeting history was a vital reference as well, and refuters showed higher interest in topics related with the rumor refuting message. Meanwhile, features such as gender, age, user labels and sentiment tendency also varied between refuters considering categories.

Keywords: rumor refuter; machine learning; nature language processing; XGBoost; feature analysis

1. Introduction

Because of the widespread popularity of social networks and mobile devices, users are able to immediately exchange information and ideas or access news reports through social media feeds such as Twitter, Facebook, or Sina Weibo [1]. However, the dark side of online information behavior should not be neglected. Due to a general lack of control, incorrect, exaggerated, or distorted information can be easily circulated throughout the networks [2]. This kind of information is defined as a rumor [3] as it does not have publicized confirmations nor official refutations. In all the controversial news stories since Twitter's inception, the rumors were found to reach more people and spread deeper and faster than the actual facts [4]. Rumors have been found to affect a country's public opinion [5], lead to economic losses [6], and even cause political consequences [7]. When the sudden crisis broke out, online rumors were even more popular, seriously disrupting social stability. For example, since January 2020, the epidemic of new coronaviruses has spread, and rumors have emerged on the Internet, causing public panic and anger and intensifying social conflicts.

Combating rumors has been a hot research area. A lot of research focuses on the rumor itself—the identification [8], spread [9–11], and influencing factors of the rumors; while deep-rooted human nature are the main factors for the viral spread of the rumor, that is, people tend to read/share tweets that confirm their existing attitudes (selective exposure), regard information that is more consistent with their pre-existing beliefs as more persuasive (confirmation bias), and prefer entertaining and incredible content (desirability bias) [4]. Existing research on the participants of the rumor is mainly aimed at influential individuals [12] in social networks. At the public level, crowd identification of rumors participants is still worth further study.

When people receive a piece of 'news', they may (1) retweet and comment at the same time, or only retweet (spreaders), (2) deny it and spread a corresponding rumor refuting message (refuters), (3) only comment on it or neglect it (stiflers). Individuals' behaviors are closely related with their attitudes [13]. Lewandowsky et al. believed that the same rumor refutation information should be changed for different opinions and angles according to the characteristics and thinking patterns of different groups of people, avoiding sensitive positions such as political positions and world views [14]. Therefore, given a rumor category, analyzing the characteristics of voluntary refuters, and identifying the special group from all rumor participants, make it possible to design targeted rumors refutation strategies based on the characteristics and thinking patterns of refuters. The application value is that the platform can consciously recommend rumor refutation information to them, even adapt the information to suit their personality. It is of great significance for expanding the acceptance of real news and suppressing the spread of rumors. Understanding the content of rumor refutation is a re-learning process, with great subjectivity and group differences. Therefore, netizens featuring analysis and crowd identification are critical to breaking through rumor governance difficulty. Recently, the rapid developments in deep learning and machine learning methods make it possible to extract and process large amounts of unstructured social media data [15–17], so as to identify different crowds and extract group features. This research topic largely remains unexplored. The only prior work was from Wang et al., who predicted social media rumor refuters only in the disaster category [18].

This study intends to reveal the features of netizens who are willing to retweet rumor refutations (refuters) without extra incentives when confronting rumor refuting messages and user features, and propose a rumor refuter crowd identification model. Five main rumor refuting microblog categories are considered that can potentially affect social stability: economics, society, disaster, politics, and military [19]. Similarities and differences of rumor refuters in these five categories are compared.

Natural Language Processing (NLP) and XGBoost are the main tools in this research. NLP is a subfield of computer science, information engineering, and artificial intelligence, and is concerned with programming computers to process and analyze large amounts of human (natural) languages data [20]. Although NLP is already a mature technology, as far as the authors know, the short text similarities and sentiment analysis have not been well-applied in combating rumors, especially associating them with rumor refuting behaviors. Baidu NLP [21] will be applied to quantify the user microblog content's sentiment (recent sentiment tendency) and similarity with original rumor refuting message (recent interests) as a value between 0 and 1, which can also be viewed as a probability. The higher the value, the higher the probability that the sentiment of the microblog is positive or the microblog content is the same as the rumor refuting message.

XGBoost [22] is a relatively new algorithm that has gained popularity due to its accuracy and robustness. XGBoost utilizes boosting, which trains each new instance to emphasize the training instances previously mis-modeled for better classification results. It is a combination of classification and regression trees (CART) [23], but re-defined the objective function with both training loss and complexity of the trees to decrease the chance of overfitting. Thus, XGBoost is a very strong model with high extensibility.

In recent years, XGBoost has been widely applied to practical problem solving [24,25]. Wang et al. have proved XGBoost was found to be the most efficient machine learning method for disaster rumor refuter identification compared with logistic regression, support vector machines, and random forest [18]. Therefore, this paper chooses XGBoost to construct the potential rumor refuters identification model.

The main contributions of this paper can be summarized as:

(1) The focus of social media users (instead of only considering influential individuals) in the rumor refutation process.

(2) Feature analysis of rumor refuters for five different categories of rumors, which can provide guidance on the personalized recommendation for social media users by accelerating the rumor refutation information dissemination.

(3) XGBoost based identification model to identify the rumor refuters and extract significant features of rumor refuters.

The remainder of this paper is organized as follows. Section 2 gives our research motivations. Section 3 shows the methods and results and Section 4 gives the discussion. Section 5 concludes the work and discusses future research applications.

2. Motivations

Research motivation lies in two aspects.

2.1. Decision-Making Support to Rumor Countermeasures

Identifying rumor refuters based on their features is quite valuable such that social media platforms can recommend rumor refuting microblogs or messages to them as this group is more likely to spread the anti-rumor information and accelerate rumor refutation [18,26]. Although there tends to be far fewer people refuting than spreading rumors [4], this ordinary refuter crowd is considerable still. Due to the potential risk of rumors, it is necessary to develop restraining countermeasures. Most identified countermeasures have been focused on blocking the rumors and spreading the truth [26]. Current practice has tended to seek to identify the influential nodes or opinion leaders to refute rumors, but has neglected the significance of the netizens willing to retweet rumor refutations without extra incentives to convince irrational followers. Therefore, from the perspective of accelerating the truth dissemination process, this paper employs feature analysis and voluntary rumor refuter crowd identification under the hypothesis that if these targeted users can be identified and taken advantage of, it is possible to gain new insights into internet rumor countermeasures.

2.2. Adapting User Features into Rumor Control

Many studies have attempted to identify how the unique features of social network users influence social media behavior. It was essential for personalized recommendation systems to detect the accurate and targeted user properties [27]. There were multiple social network identities such as microblog authors, stiflers, and retweeters. For example, some researchers collected potential author attributes such as gender, age, regional origin, and political orientation and found some feature-based differences [28]; others differentiated the features of stiflers and retweeters, and concluded that the stiflers were more concerned about social relationships and the retweeters were more driven by message content [5].

All of this prior research contributed to this paper's feature set construction. In addition, user retweet histories, status, active time, and interests also impacted retweet behavior. Hence, the similarities between the content of the target tweet and past retweeter posts [29] and users' subjective feelings were also determining factors [30]. However, few works have been done in adapting those user analyses into rumor control and refuter identification. Previous investigations have involved social media platforms with different user structures (i.e., Twitter and Facebook), but the conclusions could not be generalized to microblog users. Overall, few studies on retweeter attributes have commented on the distinctions between the different original microblog types; therefore, this paper analyzes the refuter features based on anti-rumor classifications.

3. Methods and Results

In this section, we present the methods and relevant results in detail. The overall framework of the methods is shown in Figure 1. Firstly, the data are collected and cleaned. Then, the gender and label frequency comparisons between refuters of five categories are made, which form a rough refuter portrait. Thirdly, sentiment analysis and short text similarity analysis are made. If there are missing values in microblogs/label information/verified information/signature, a value of 0.5 will be assigned. Based on the trained XGBoost classification model, the refuter feature analysis is conducted.

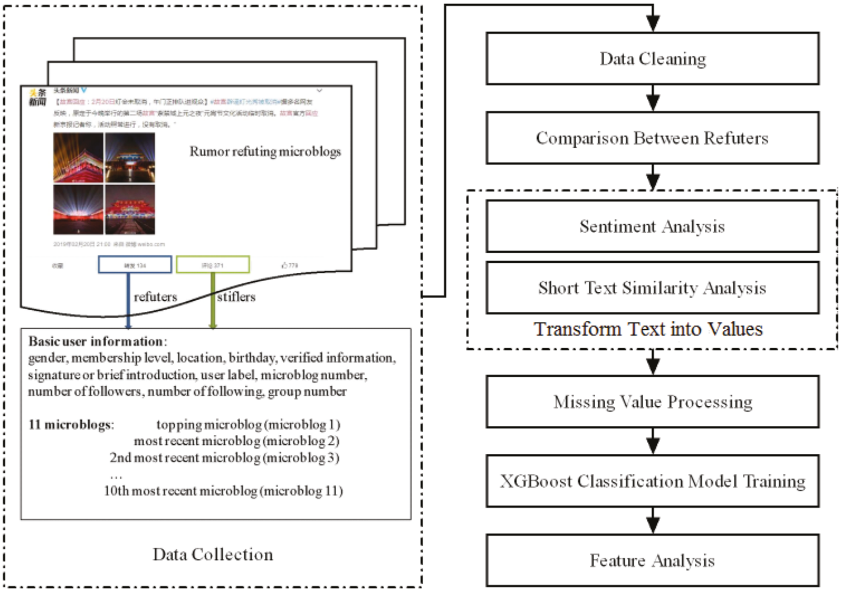

Rumor refuting microblogs
refuters
stiflers
Basic user information:
gender, membership level, location, birthday, verified information, signature or brief introduction, user label, microblog number, number of followers, number of following, group number
11 microblogs: topping microblog (microblog 1)
most recent microblog (microblog 2)
2nd most recent microblog (microblog 3)
…
10th most recent microblog (microblog 11)
Data Collection
Data Cleaning
Comparison Between Refuters
Sentiment Analysis
Short Text Similarity Analysis
Transform Text into Values
Missing Value Processing
XGBoost Classification Model Training
Feature Analysis

Figure 1. The overall framework of the methods.

3.1. Data Collection

Sina Weibo is a Chinese microblogging (Weibo) website and is one of the most popular social media platforms in China with 431 million active monthly users in 2019. Different from Wechat, which only allows a user to post to certified friends, Sina Weibo is the Chinese equivalent of Twitter as it has a wider, more open dispersal. Therefore, crawling microblogs on Sina Weibo has a high research value for rumor propagation or rumor refutation spread analyses. All of the anti-rumor microblogs with a retweet/comment amount larger than 100 were collected from October 2018 to April 2019 using a web crawler.

This paper only takes the anti-rumor microblogs verified, confirmed, and announced by official accounts (police accounts, government agency accounts, and authoritative media accounts) into considerations. Therefore, the refuters discussed in this paper are those who deliberately spread official accounts' rumor refutation information. As shown in Figure 2, the collected anti-rumor microblogs were classified into five categories based on content [18]; economics, society, disaster, politics, the military, all of which were the common rumors on social media platforms and could result in societal damage. The economic category contained business and entrepreneurial information; the society category covered rumors about social public affairs; the disaster category consisted of distorted information on natural and man-made disasters; the politics category comprised false political messages mainly involving certain political figures, groups, or specific policies; and the military category included rumors about national defense or military affairs.

These five main categories had a total of 106 anti-rumor microblogs, of which 45 were related with the society, 31 with economics, three with the military, 20 with politics and seven with disaster, with a total of 58,807 user samples. There were far more stiflers than refuters collected because the task of identifying the refuters from the population was inherently an imbalanced classification problem. As this research was simulating the refuter identification process and examining the validity of XGBoost model, testing on a small data subset was considered powerful enough to examine the algorithm's performance [31].

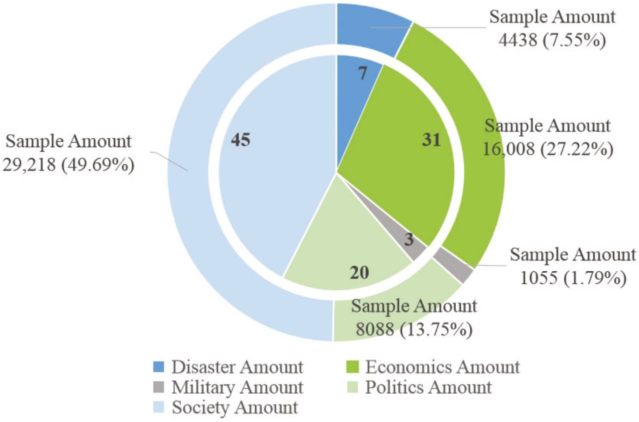


Figure 2. Microblog and sample quantities.

The users' most recent concerns were strongly associated with their most recent microblogs and our previous work found that the 11 most recently posted microblogs (topping microblogs are included) were reliable predictors in disaster rumor refuter prediction [18]. Except for a few users who had less than 11 microblogs since registration, 11 microblogs were extracted, i.e., the topping microblog (the sticky microblog) and 10 most recently posted microblogs. Although the topping microblogs might not have been recently posted, they were able to reveal the overall attitude of the users to some extent. Basic user information; gender, membership level, location, birthday, verified information, signature or brief introduction, user label, microblog number, number of followers and numbers following, and group numbers for each user; were extracted.

As the aim of this research was to identify the social media rumor refuter features, information was mined for two groups of people: refuters from the retweet lists and stiflers from the comment lists. Stiflers consist of the commenters and the users who only view the rumor refuting message. Due to the inaccessibility of the viewer list, only those commenters were treated as stiflers. Although both of refuters and stiflers had viewed the rumor refuting microblogs, the responses were quite different.

3.2. Comparison between Refuters

3.2.1. Gender

Figure 3 compares the gender differences for different categories. The gender gap was particularly large in the military and political fields, with the number of male refuters being nearly twice as many as females (roughly 150 men vs. 74 women and 1621 men vs. 888 women, respectively). In contrast, in the economic, disaster and society categories, there were only minor gender differences. In 2018, male users made up 57% of total Weibo users [32]. The results are correspondent with the Weibo user gender ratio.

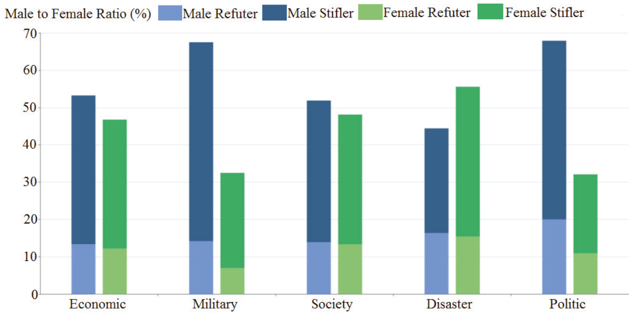


Figure 3. Male to female ratio in database.

3.2.2. Label Frequency

A user portrait analysis was conducted based on the refuter label information. From the word frequency count, it was possible to roughly depict the refuter features and preferences.

As can be seen in Table 1, economics-related rumor refuters showed high interest in IT, Dig, and investment, with most being young practitioners in the internet or finance industries. The economic-related and politics-related rumor refuters had some common interests (i.e., military, investment, Finance and IT, and Dig), and could be the same group of people. For the military-related rumor refuters, the label "military" was third ranked, with interest also being shown in design and history. The society-related and disaster-related rumor refuters were also both interested in education, with the former group having a specific "campus" label and the latter group having a specific "employment" label. Based on this information, we infer that, for these groups, college students should account for a relative large proportion of refuters.

Table 1. Label frequency for the different rumor refuters.

Economics		Military		Society		Disaster		Politics	
Label	**Rank**	**Label**	**Rank**	**Label**	**Rank**	**Label**	**Rank**	**Label**	**Rank**
IT&Dig	5	**Military**	3	IT&Dig	8	IT&Dig	10	IT&Dig	6
Invest	12	Invest, Finance	6	**Reading**	14	Invest	11	Finance	8
Military	16	**News**	11	**Fashion**	19	Finance	15	**News**	9
Finance	17	**Design**	12	**Education**	20	**Education**	18	**Military**	11
Reading	20	**History**	15	**Campus**	21	**Employment**	19	Invest	18

3.3. *Rumor Refuters Identification*

A crowd identification process was applied in two steps.

Step 1. Convert the textual content into numerical values.

For rumors that are linked to specific geographical locations, we derived the locations from the original rumor texts. Then, comparing the locations where the rumor "took place" with the locations each user from, 1 would be assigned for the location feature if the user was in the same province as the rumor, and 0 otherwise.

Baidu's AipNLP [21], which is regarded as the most advanced Chinese text analysis technique, was applied to convert the textual content into numerical values. Then, the similarities between the user labels, the verified information, the signature, the most recent 11 microblog (including the topping microblog) contents, and the rumor refuting microblogs were transformed into values between 0 and 1. For the sentiment analysis, the emotional inclinations of the user signatures and the most recent 11 microblog contents were also converted into values between 0 and 1. The processed variables are listed in Table 2.

Additional implementations were applied to the variables to ensure the classification results were more valid and reliable:

(1) The corresponding rumor refuting microblogs were deleted if they were one of the 11 most recent microblogs from the user.

(2) A value of 0.5 was assigned to the microblogs/label information/verified information/signature sentiment or short text similarity analysis if the text was missing.

(3) Words irrelevant to the content of the text but that significantly influenced the result of the sentiment analysis, such as “Comment”, “Like”, and “Collect”, were removed.

Table 2. Variables for refuter identification.

Variables	Variable Descriptions
R	Dependent variable. Whether the user retweeted the rumor refuting microblog. 1 for yes and 0 for no.
G	Gender of the user set. 1 for male and 0 for female.
ML	Membership Level. Explains the user devotion and activity to some extent.
L	Location. The provincial level location of the user. 1 if the user was in the same province as the rumor; 0 otherwise.
A	Age. The age of the users. If the value was missing, we interpolated the average age in that category.
LSm	Similarity between the label information and the rumor refuting microblog ranging from 0–1; the larger the value, the more similar the two texts.
VISm	Similarity between the verified information and the rumor refuting microblog ranging from 0–1; the larger the value, the more similar the two texts.
SSe	Sentiment of the Signature or a Brief Introduction of the user ranging from 0–1; the larger the value, the more positive the attitude.
SSm	Similarity between the Signature or Brief Introduction of the user and the rumor refuting microblog ranging from 0–1; the larger the value, the more similar the two texts.
NOM	Number of Microblogs the user has already posted.
NOU	Number of Users the user has followed.
NOF	Number of Followers the user has.
NOG	Number of Groups the user has classified as friends.
MSe (1–11)	Sentiment of the ith microblog of the user ranging from 0–1; the larger the value, the more positive the attitude.
MSm (1–11)	Similarity between the ith microblog of the user and the rumor refuting microblog ranging from 0–1; the larger the value, the more similar the two texts.

Step 2. The XGBoost model was utilized for refuter identification in the different categories.

In this research, the XGBoost model got a bi-classification task. Thus, people who forwarded the anti-rumor microblog was treated as rumor refuters and labeled 1, people who only commented but not retweeted was treated as rumor stiflers and labeled 0.

The samples were randomly divided into two parts, 80% for training the XGBoost model and the other 20% for testing the effect of the trained model. Two criteria; the F1 Score [33] and the AUC [34]; were applied for the classification result evaluation (as shown in Table 3). F1 score is a measure of a test’s accuracy. It considers both the precision and the recall so that it is practical in classification tasks [33]. AUC is the probability of ranking the positive sample forward the negative sample whenever a positive and a negative samples are randomly selected. The higher the AUC, the better the classification result is [34].

With learning rate of 0.05, max_depth of 10, subsample value of 0.8, scale_pos_weight corresponding to the proportion of positive and negative samples, and keeping all the other parameters as default, the model is trained by Python Xgboost package (num_boost_round = 300 and early_stopping_rounds = 50).

The efficiency of the XGBoost model can also be impacted by the number of samples. Therefore, for the disaster, economic, political, and societal categories and all samples, the amount of samples (randomly selected each time and not applied to the military category because there were only

1055 samples) was gradually increased to examine the influence of the number of samples on the classification results. During this process, different samples were applied for robustness testing and to determine the relationship between sample quantity and the F1 Score/AUC Score when the XGBoost model was applied, with the overall aim being to determine the number of samples needed to obtain a stable, available F1 Score/AUC Score.

The feature importance was also ranked using the XGBoost model to determine the most important refuter crowd identification features for the different rumor categories. For those most important features, *t*-tests were implemented, to identify which individual feature, for instance, number of microblogs or number of followers, was significantly different between refuters and stiflers.

Because the F1 Score and AUC curves were similar, for better observation, only the F1 Score curve was drawn. As it is shown in Figure 4, starting with 500 randomly selected samples, the F1 Score was observed to gradually increase and then, as sample quantity increased in all categories, it became stable at around 0.75 (except for the military category that had only 1055 samples and F1 and AUC scores of 0.65). Even when all sample types were included, the observations remained the same.

Table 3. AUC and F1 Scores for each category and all samples.

Index	Military	Disaster	Politics	Economics	Society	All
AUC	0.6501	0.7404	0.7350	0.7356	0.7304	0.7356
F1 Score	0.6501	0.7488	0.7470	0.7511	0.7446	0.7490

Therefore, it was concluded that, when plenty of data were provided, the XGBoost model was effective in identifying rumor refuters irrespective of the rumor category differences.

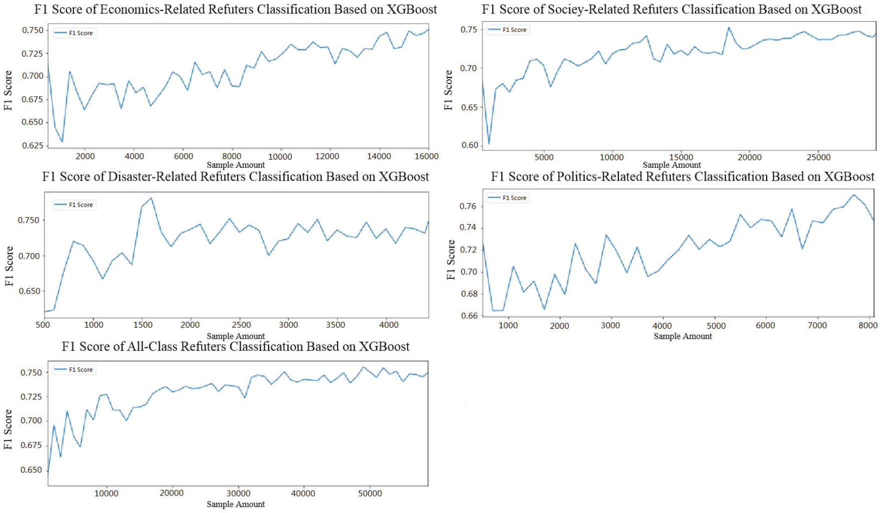


Figure 4. Refuter classification results based on XGBoost.

3.4. Feature Analysis between Refuters and Stiflers

The XGBoost model also provided feature importance rankings (see Figure 5). Except for the disaster and political categories, gender was found to be the least important feature in the XGBoost classifications. However, the MSe11 and MSm11 (regard the topping microblog as the 1st microblog, and MSe11 and MSm11 refer to the sentiment value and similarity with the origin rumor refuting microblog of the 10th most recent microblog respectively) appeared to have the most important features for all categories.

It was therefore proposed that, if the user had the topping microblog and an emotional inclination and similarity to the original microblog, there would be an influence on the classification judgement. Therefore, samples with MSe11 and MSm11 not equal to 0.5 (i.e., samples with topping microblog) were extracted and their Mse1 and MSm1 were tested and the results are shown in Table 4. However, there were no significant differences between the refuter and stifler values for their MSe1 and MSm1 in the political-related, disaster-related and military-related categories. The economics-related and societal-related rumor refuter values for MSm1 were lower than those of the stiflers. There were no significant differences in the MSe1 values between the stiflers and refuters in the economics-related category but, for the societal-related category, the refuters' MSe1 values were higher than those of the stiflers.

The NOM and NOF were also found to be very important features. The *t*-test results in Table 4 found that the rumor refuter NOM and NOF values were somewhat higher than those for the stiflers, which indicated that refuters could be more active. The ML (another measurement of user activity) was also somewhat higher for the refuters in the economics, politics, and societal categories.

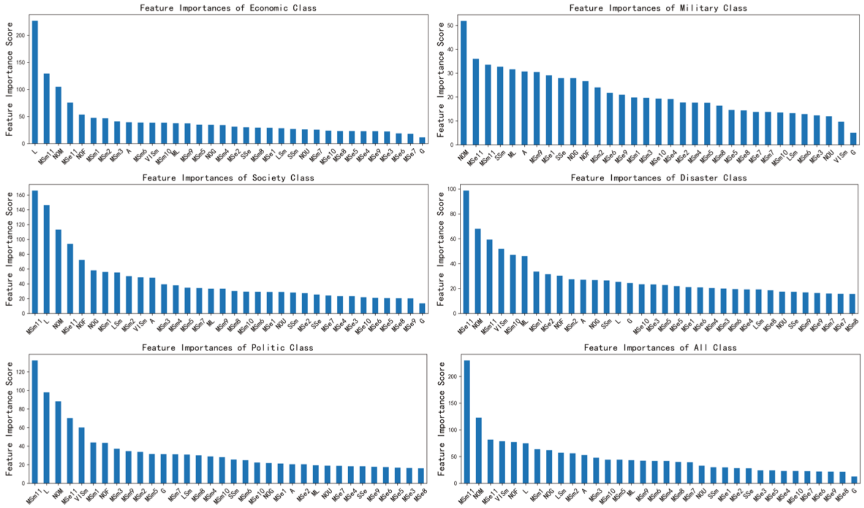


Figure 5. Feature importance in the different categories.

Table 4. *T*-test results for the rumor refuters and stiflers.

	Economic	Military	Societal	Disaster	Political
ML	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
NOM	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
NOF	**0.000**	**0.003**	**0.000**	**0.000**	**0.000**
VISm	**0.000**	**0.001**	**0.000**	**0.000**	**0.000**
LSm	**0.000**	**0.033**	0.335	**0.000**	**0.000**
SSm	**0.000**	**0.000**	**0.000**	**0.027**	0.815
MSe1	0.501	0.526	**0.001**	0.792	0.215
MSm1	**0.023**	0.738	**0.001**	0.540	0.439
$\overline{\text{MSe}}$	**0.001**	0.526	0.512	0.818	0.240
$\overline{\text{MSm}}$	**0.000**	**0.003**	**0.000**	**0.000**	**0.000**

Note: (the value is shown in bold if there were significant differences between the refuters and stiflers under a 95% confidence level, "___" means that the refuter value was lower than that of the stiflers; for users with a topping microblog, the MSe1 and MSm1 refuter and stifler values were compared.).

Except for the LSm in the societal category and the SSm in the political category, there were significant differences found for the VISm, LSm, and SSm between the refuters and stiflers at a 95% confidence level. For the economic, disaster, and military categories, the VISm and LSm of the refuters were significantly lower than those of the stiflers, while the refuters had higher SSm values. Similarly, for the societal category, the VISm of the refuters was significantly lower than that of stiflers, while the refuters had higher SSm values. In contrast, in the political category, the VISm and LSm of refuters were significantly higher than those of the stiflers.

The average values for the MSe (1–11) and the MSm (1–11) for the refuters and the stiflers in the 5 main categories were calculated and denoted $\overline{\text{MSe}}$ and $\overline{\text{MSm}}$. As shown in Table 4, there were no significant differences found between the refuters and the stiflers for the $\overline{\text{MSe}}$ at a 95% confidence level, except for the economic category (the $\overline{\text{MSe}}$ of refuters was significantly lower than that of stiflers). The $\overline{\text{MSm}}$ of the refuters in all five categories, however, was higher than that of the stiflers, which indicated that the average short text refuter similarity degrees with the original rumor refuting microblogs were significantly higher than those of the stiflers.

According to Table 5, at the 95% confidence level, in the disaster, economic and society related rumor refuting microblogs, correlations were confirmed between user behavior (refute/stifle) and user location (whether in the same province in which the rumor-related event occurred); however, in the political category, no correlations were found.

Table 5. Chi-square test of contingency results between user behavior and user location.

Category	Degree of Freedom	Chi-Value	*p*-Value
Economic	1	12.6312	0.0004
Societal	1	13.6454	0.0002
Disaster	1	65.7010	0
Politic	1	2.8048	0.094
Category & Behavior	4	142.3592	0

As shown in Table 6, for users in the same location as the rumor refuting microblogs, the refuters were found to be less likely to retweet disaster, economic or society related rumor refutation information, with only 21.56%, 15.67%, and 23.77% of total viewers in the same province. One possible explanation is that these refuters know better about the local situation and do not feel the urge to spread truths. Therefore, the social media platform can recommend disaster, economic, or society related anti-rumor information to users not in the same location as the rumor refuting microblogs.

Table 6. Refuter proportions in the same province as the anti-rumor microblogs.

Category	Disaster	Economic	Military	Politics	Society
Refuters in the same province	224	39	0	42	498
Stiflers in the same province	815	210	0	66	1597
Refuter proportion in the same province	21.56%	15.67%	-	38.89%	23.77%
Refuters not in the same province	1190	4062	224	2467	7466
Stiflers not in the same province	2209	11,697	831	5513	19,657
Refuter proportion not in the same province	35.01%	25.77%	21.23%	30.91%	27.53%

4. Discussion

Based on feature analysis of users with different social media behavior, this study sought to identify the potential voluntary rumor refuter, and utilize them with the anti-rumor countermeasure: truth propagation and targeted immunization. Because of the growing popularity of social media and the availability of complete user information, it is possible to accurately obtain user features and therefore easier to identify the potential refuters. Thus, personalized recommendation services could be provided to trigger the spread of the truth, and thus enhance rumor refutation.

Although previous works have explored the features of retweeters, there have been few studies on utilizing these findings to combat rumor spread. This paper extended the scope of current studies, instead of studying the general features of retweeters or the opposite group, rumor spreaders, it focused on refuters and specified them with five main rumor categories that can affect social stability. Although both rumor spreaders and rumor refuters have the same behavior—retweet, their features were different. In contrast with the conclusion of Vosoughi et al. [4], in which the rumor spreaders were found to have less followers, it was observed that the rumor refuters had a greater number of microblogs and followers; i.e., they were more active. However, this result could be partially explained by Zhang et al. [35] that social relationship and message content were noticeable driven factors of retweeting behavior. Our findings were also in line with the literature indicating that users mainly retweet to remind others and express themselves, and retweetability is closely related to the number of followers and tweet contents' information and value [36]. It can be recognized that, when user got more followers, they tended to be more cautious with their microblog contents. Thus, retweeted messages that seem more reliable, and rumor refuting messages released by authoritative media could be one of those.

Except for the economic category, there were no significant general sentiment tendency differences between stiflers and refuters. However, the microblog contents and signature contents (except for the political type) of refuters got higher similarity with the original rumor refuting message, and this result was consistent with Luo et al. [29] and Macskassy and Michelson [37]. On the contrary, the similarity between rumor refuting message and verified information and user label were generally lower for refuters (except for the political type), which indicated that the circles and occupations of users were not seriously constant with their daily interests on the social media. Meanwhile, refuters tended to gather in more developed regions.

There were specific rumor refuter feature variations in the microblog categories that had not been previously detected. The politic and military related rumor refuters were generally older and many of them showed interests in finance and investment. Oppositely, the younger ones were more likely to be economic, society, and disaster related rumor refuters. Many of them showed interests in IT&Dig, reading, fashion, education, and employment, and those labels matched their age well. Users in the same province with the rumor seemed less likely to retweet the rumor refuting message. This phenomenon could be explained by the third person effects [38]. On the one hand, the more negative the event was, the more obvious the third person effects were. Due to peoples' underlying sense of superiority and confidence, they unconsciously believe that negative content would exert greater impacts on others than themselves and thus lead to their retweet behaviors to convey information to others (it was also why this phenomenon was most obvious in disaster related rumors). On the other hand, the effectiveness of third person effect is strongly influenced by the geographical distance between the receiver and information source, implying that the farther the receiver is from the information source, the stronger the third-person effect. Therefore, people in other locations thought that retweeting right message was urgent and important, considering those people with both long social and geographical distance would be significantly influenced by media content.

However, the small microblog sample size may have influenced the study's validity to certain extent, and the study was also limited by some of the basic variables that were extracted to characterize the refuter profiles. As the issue of user features has always been intriguing and could be explored from various dimensions, it is expected that, in the future, a wider range of features will be identified in future works to more comprehensively model rumor refuters such as ethnicity, personal preferences, active time, and sociolinguistic features. More empirical studies are also needed to investigate the usefulness and feasibility of the method developed in this paper on other social media platforms such as Wechat, Facebook, and Twitter so that it can be incorporated into active applications.

An additional uncontrolled factor is a difficulty in accurately identifying rumors/anti-rumor on Weibo. In terms of the Chinese legal framework, rumors are generally fake news. This definition emphasizes the deviation from the truth and the fact. From the perspective of mass communication, a rumor is the statement or piece of news that is deliberately made up out of thin air. The malicious motives behind the information source might also be considered. In addition to the rumor itself, there are other forms of information filled up with Weibo, such as uncertain information and speculative information. It can be seen that there is no unified definition of rumors from different academic perspectives, and there are no clear judgment criteria for rumors, so, in practice, many difficulties and problems are unavoidable in the identification of rumors.

The principal purpose of countering the rumors is to filter the literal meaning of rumors, dig into once-hidden problems behind the rumors, and solve the underlying deep-seated social problems reflected, effectively responding to the social anxiety. Given that the main body in China to deal with rumors, solve social problems, and take targeted actions is mostly government agencies, this paper takes whether the false information/refutes of rumors posted by mainstream official accounts (such as police department accounts, government accounts and authoritative media accounts) as the criteria for recognizing and identifying rumors/anti-rumor, so as to maximize the distinction between truth and rumor. Such criteria might still lead to bias in rumors/anti-rumor judgments. Further research might add more dimensions and standards to search and identify rumors/anti-rumor on Weibo, for instance, taking the scientificity, social influence and the poster’s subjective intention and other aspects of the web message into the comprehensive consideration.

5. Conclusions and Future Work

The purpose of the current study was to determine the association between user features (including sentiment tendency, recent interests, gender, geographic distributions, age structure, and label frequency) and their refuting behaviors and so as to identify the rumor refuters, and deal with the dark side of online information behavior by accelerating the rumor refutation information dissemination.

The findings shown in Table 7 reveal some general features of refuters as well as variations between refuters considering different rumor categories: (1) there were more male refuters than females, especially in the politics and military categories; (2) rumor refuters of all categories were found to be highly concentrated in East, North and South of China, and particularly in provinces with first-tier cities; (3) when users were from the same geographic locations as the refutation microblogs, they were less inclined to retweet economic, societal and disaster related rumor refutation microblogs; (4) refuters were mainly aged between 18 and 40, with the refuters in the politics and military categories being somewhat older than those in the economic, society, and disaster-related categories; and (5) the political and society related rumor refuters tended to follow and post relevant information more frequently, which was shown by their higher $\overline{\text{MSm}}$.

On the other hand, as it is shown in Table 8, there were significant differences between refuters and stiflers: (1) rumor refuters were found to be more active with higher NOM and NOF; (2) the ML was comparatively higher in the economic, political and societal categories; (3) in general, the refuters’ VISm and LSm were significantly lower than the stiflers (except for the LSm in the societal category), but their SSm was higher; however, the refuters in the political category were found to have higher VISm and LSm than the stiflers and there were no significant differences between the SSm of rumor refuters and stiflers; (4) economic related rumor refuters had less positive microblog content sentiment, but the refuters tended to have higher $\overline{\text{MSm}}$ in all categories.

Provided that there was an adequately large amount of data, the XGBoost model was broadly applicable in identifying the refuters, regardless of differences in the rumor categories.

This paper only takes the anti-rumor posts verified, confirmed, and announced by official accounts into consideration, but there is still a small chance that a rumor refuted by the platform turn out to be true eventually. In the future, we plan to examine the refuter characteristics in a wider range of microblog samples, hopefully covering the possible bias with large data. In addition, we will consider more personalized and individualized features beyond just demographic attributes to more precisely identify the refuter crowd. Analysis on influence and power of refuters is also a focus of future research.

Table 7. General features of the rumor refuters.

Category	Common Feature	Gender	Age	Special Interests	Role	Microblog Content	
						Sentiment	Similarity
Economic	Aged: 18–40 ML: 0.89 (mean) NOM: 1363 (median) NOF: 169 (median) NOU: 303(median) Regions: East, North & South China; province with 1st-tier cities (particularly Beijing and Shanghai)	balanced	18.15–37.36	IT & Dig, Invest, Military, Finance	younger practitioners in the internet & finance industries	mean: 0.588 (higher than politic & society)	mean: 0.656
Military		more male	20.14–43.94	Military, Design, History, News	older practitioners in the news industries	mean: 0.575	mean: 0.703
Society		balanced	17.99–39.79	Education, Campus, Reading, Fashion	college students or fresh graduates	mean: 0.558	mean: 0.667 (higher than economic and disaster)
Disaster		balanced	18.08–40.68	Education, Employment	college students or fresh graduates	mean: 0.569 (higher than politic & society)	mean: 0.666 (higher than economic)
Politic		more male	20.32–43.06	IT & Dig, News, Military, Finance	older practitioners in the internet & finance industries	mean: 0.557	mean: 0.674 (higher than economic, society, disaster)

Table 8. Features of the rumor refuters compared with the rumor stifler.

Category	Common Feature	Same Province	ML	Custom Info			Microblog Content
				VISM	LSM	SSM	Sentiment
Economic	Higher: NOM, NOF, $\overline{\text{MSm}}$	less likely to refute	0–2.65 (higher)	0.41–0.55 (lower)	0.14–0.54 (lower)	0.28–0.60 (higher)	relatively less positive
Military		no significant difference	0–1.76 (lower)	0.43–0.55 (lower)	0.12–0.54 (lower)	0.32–0.60 (higher)	no significant difference
Society		less likely to refute	0–2.59 (higher)	0.41–0.55 (lower)	no significant difference	0.28–0.60 (higher)	no significant difference
Disaster		less likely to refute	0–2.55 (lower)	0.41–0.55 (lower)	0.18–0.56 (lower)	0.33–0.63 (higher)	no significant difference
Politic		no significant difference	0–2.76 (higher)	0.41–0.55 (lower)	0.15–0.55 (higher)	no significant difference	no significant difference

Author Contributions: Conceptualization, S.W., Z.L., and Y.W.; Data Curation, S.W., Q.Z., and Y.W.; Formal Analysis, S.W., Z.L., Y.W., and Q.Z.; Investigation, S.W. and Y.W.; Methodology, S.W., Z.L., Q.Z., and Y.W.; Supervision, Z.L.; Validation, S.W., Z.L., Y.W., and Q.Z.; Writing—Original Draft, S.W., Z.L., Y.W., and Q.Z.; Funding acquisition, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the China Postdoctoral Science Foundation Funded Project (Grant No. 2017M612983), Chengdu Philosophy and Social Science Planning Project (Grant No. 2019L40), and the Fundamental Research Funds for the Central Universities (Grant No. SCUBS-PY-202017).

Acknowledgments: We thank the editors and any reviewers for their helpful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, C.; Xin, Y.; Li, X.; Yang, Y.; Chen, Y. A Heterogeneous Ensemble Learning Framework for Spam Detection in Social Networks with Imbalanced Data. *Appl. Sci.* **2020**, *10*, 936. [CrossRef]
2. Mihailidis, P.; Viotty, S. Spreadable Spectacle in Digital Culture: Civic Expression, Fake News, and the Role of Media Literacies in "Post-Fact" Society. *Am. Behav. Sci.* **2017**, *61*, 441–454. [CrossRef]
3. Difonzo, N.; Bordia, P.; Rosnow, R.L. Reining in rumors. *Organ. Dyn.* **1994**, *23*, 47–62. [CrossRef]
4. Vosoughi, S.; Roy, D.; Aral, S. The spread of true and false news online. *Science* **2018**, *359*, 1146–1151. [CrossRef] [PubMed]
5. Ramos, M.; Shao, J.; Reis, S.D.S.; Anteneodo, C.; Andrade, J.S.; Havlin, S.; Makse, H.A. How does public opinion become extreme. *Sci. Rep.* **2015**, *5*, 10032. [CrossRef]
6. Syrian Hackers' Break into Associated Press' Twitter Account and 'Break News' that Explosions at White House have Injured Obama-Sending DOW Jones Plunging 100 Points. Available online: goo.gl/NSliQP (accessed on 20 December 2019).
7. Humprecht, E. Where "fake news" flourishes: A comparison across four Western democracies. *Inf. Commun. Soc.* **2019**, *22*, 1973–1988. [CrossRef]
8. Liu, Y.; Jin, X.; Shen, H. Towards early identification of online rumors based on long short-term memory networks. *Inf. Process. Manag.* **2019**, *56*, 1457–1467. [CrossRef]
9. Qian, Z.; Tang, S.; Zhang, X.; Zheng, Z. The independent spreaders involved SIR Rumor model in complex networks. *Phys. A Stat. Mech. Appl.* **2015**, *429*, 95–102. [CrossRef]
10. Xia, L.-L.; Jiang, G.-P.; Song, B.; Song, Y. Rumor spreading model considering hesitating mechanism in complex social networks. *Phys. A Stat. Mech. Appl.* **2015**, *437*, 295–303. [CrossRef]
11. Zhang, Y.; Xu, J. A Rumor Spreading Model considering the Cumulative Effects of Memory. *Discret. Dyn. Nat. Soc.* **2015**, *2015*, 1–11. [CrossRef]
12. Goel, S.; Anderson, A.; Hofman, J.; Watts, D.J. The structural virtuality of online diffusion. *Manag. Sci.* **2015**, *62*, 180–196. [CrossRef]
13. Almodarresi, S.M.A.; Tabatabainasab, S.M.; Garabollagh, H.B.; Mohammadi, F. Does citizenship behavior have a role in changing attitude toward green products. *Int. J. Manag. Sci. Eng. Manag.* **2019**, *14*, 284–292. [CrossRef]
14. Lewandowsky, S.; Ecker, U.K.H.; Seifert, C.M.; Schwarz, N.; Cook, J. Misinformation and Its Correction: Continued Influence and Successful Debiasing. *Psychol. Sci. Public Interest* **2012**, *13*, 106–131. [CrossRef]
15. Gholami, P.; Hafezalkotob, A. Maintenance scheduling using data mining techniques and time series models. *Int. J. Manag. Sci. Eng. Manag.* **2018**, *13*, 100–107. [CrossRef]
16. Iglesias, C.A.; Moreno, A. Sentiment analysis for social media. *Appl. Sci.* **2019**, *9*, 5037. [CrossRef]
17. Alotaibi, S.; Mehmood, R.; Katib, I.; Rana, O.; Albeshri, A. Sehaa: A big data analysis tool for healthcare symptoms and diseases detection using twitter, apache spark, and machine learning. *Appl. Sci.* **2020**, *10*, 1398. [CrossRef]
18. Wang, S.; Li, Z.; Wang, Y.; Zhang, Q. Machine Learning Methods to Predict Social Media Disaster Rumor Refuters. *Int. J. Environ. Res. Public Health* **2019**, *16*, 1452. [CrossRef] [PubMed]
19. Wang, G. Dealing with rumors and their control methods from the perspective of communication. *J. Commun.* **1991**, *1*, 41–56. (In Chinese)
20. The History of Machine Translation in a Nutshell. Available online: http://hutchinsweb.me.uk/Nutshell-2005.pdf (accessed on 22 December 2019).
21. Baidu's Aip NLP. Available online: https://pypi.org/project/baidu-aip/ (accessed on 26 December 2019).
22. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
23. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. *Classification and Regression Tree*; Chapman & Hall: New York, NY, USA, 1984.
24. Zheng, C.Y.; Pestilli, F.; Rokem, A. Deconvolution of High Dimensional Mixtures via Boosting, with Application to Diffusion-Weighted MRI of Human Brain. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Nice, France, 2014; pp. 2699–2707.

25. Luo, H.; Schapire, R.E. A Drifting-Games Analysis for Online Learning and Applications to Boosting. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Nice, France, 2014; pp. 1368–1376.
26. Wen, S.; Jiang, J.; Xiang, Y.; Yu, S.; Zhou, W.; Jia, W. To shut them up or to clarify: Restraining the spread of rumors in online social networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 3306–3316. [CrossRef]
27. Khalili-Damghani, K.; Abdi, F.; Abolmakarem, S. Solving customer insurance coverage recommendation problem using a two-stage clustering-classification model. *Int. J. Manag. Sci. Eng. Manag.* **2019**, *14*, 9–19. [CrossRef]
28. Rao, D.; Yarowsky, D.; Shreevats, A.; Gupta, M. Classifying latent user attributes in twitter. In Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents, Toronto, ON, Canada, 26–30 October 2010; pp. 37–44.
29. Luo, Z.; Osborne, M.; Tang, J.; Wang, T. Who will retweet me? Finding retweeters in twitter. In Proceedings of the 36th international ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 28 July–1 August 2013; pp. 869–872.
30. Sun, J.; Wang, G.; Cheng, X.; Fu, Y. Mining affective text to improve social media item recommendation. *Inf. Process. Manag.* **2015**, *51*, 444–457. [CrossRef]
31. Petrak, J. Fast subsampling performance estimates for classification algorithm selection. In Proceedings of the ECML 2000 Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection and Method Combination, Barcelona, Spain, 31 May–2 June 2000; pp. 3–14.
32. 2018 Weibo User Development Report. Available online: http://data.weibo.com/report/reportDetail?id=433 (accessed on 19 December 2019).
33. Powers, D.M.W. Evaluation: From precision, recall and F-measure to ROC. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
34. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2005**, *27*, 861–874. [CrossRef]
35. Zhang, J.; Liu, B.; Tang, J.; Chen, T.; Li, J. Social influence locality for modeling retweeting behaviors. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013; pp. 2761–2767.
36. Suh, B.; Hong, L.; Pirolli, P.; Chi, E.H. Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network. In Proceedings of the IEEE International Conference on Social Computing, Minneapolis, MN, USA, 20–22 August 2010.
37. Macskassy, S.A.; Michelson, M. Why do people retweet? Anti-homophily wins the day. In Proceedings of the 5th International AAAI Conference on Weblogs and Social Media, Barcelona, Spain, 17–21 July 2011.
38. Jang, S.M.; Kim, J.K. Third person effects of fake news: Fake news regulation and media literacy interventions. *Comput. Hum. Behav.* **2018**, *80*, 295–302. [CrossRef]

applied sciences

Article

The Efficiency of Social Network Services Management in Organizations. An In-Depth Analysis Applying Machine Learning Algorithms and Multiple Linear Regressions

Luis Matosas-López [1,*] and Alberto Romero-Ania [2]

[1] Department of Financial Economics, Accounting and Modern Language, Rey Juan Carlos University, Paseo Artilleros s/n, 28032 Madrid, Spain

[2] Department of Applied Economics, Rey Juan Carlos University, Paseo Artilleros s/n, 28032 Madrid, Spain; alberto.romero@urjc.es

* Correspondence: luis.matosas@urjc.es

Received: 25 June 2020; Accepted: 24 July 2020; Published: 27 July 2020

Abstract: The objective of this work is to detect the variables that allow organizations to manage their social network services efficiently. The study, applying machine learning algorithms and multiple linear regressions, reveals which aspects of published content increase the recognition of publications through retweets and favorites. The authors examine (I) the characteristics of the content (publication volumes, publication components, and publication moments) and (II) the message of the content (publication topics). The research considers 21,771 publications and thirty-nine variables. The results show that the recognition obtained through retweets and favorites is conditioned both by the characteristics of the content and by the message of the content. The recognition through retweets improves when the organization uses links, hashtags, and topics related to gender equality, whereas the recognition through favorites increases when the organization uses original tweets, publications between 8:00 and 10:00 a.m. and, again, gender equality related topics. The findings of this research provide new knowledge about trends and patterns of use in social media, providing academics and professionals with the necessary guidelines to efficiently manage these technologies in the organizational field.

Keywords: machine learning algorithms; multiple linear regression; support vector machines; SVM; management; social network services

1. Introduction

The widespread use of the Internet has prompted numerous changes in recent decades. The Internet has transformed all sectors of the economy and society as a whole. In this sense, social network services are one of the best examples of how technology has changed our behavior patterns.

In 2020, the number of Internet users in the world is 4.54 billion, with an average penetration of social network use of 49% [1]. This global aggregate penetration datum obviously varies between countries. Thus, for example, the percentage in India is 29%, in Germany 45%, in the United States (US) 70%, and in South Korea 87% [1]. In the particular case of Spain, two-thirds of the population are regular users of platforms such as Facebook and Twitter [2].

The level of penetration of these technologies has transformed the way people interact with their environment, to the point of making it necessary to create a descriptive term for the typical user of these platforms: "media prosumer". Some authors describe the media prosumer as the subject capable of taking center stage, producing and consuming information in the net [3]. Others define the media prosumer as that user who actively assumes the role of the communication channel, taking advantage of it to become a recommender on different topics [4].

One of the approaches used by researchers to examine the behavior of the media prosumer on social network services is that of the uses and gratifications theory. Although this theory was initially developed to describe how audiences interact with mass media, such as radio, the press, or television [5,6], the power of these technologies to propagate information to large audiences, as traditional media do, makes the uses and gratifications theory especially suitable for contextualizing research in this field.

The conceptual framework defined by the uses and gratifications theory allows researchers to explore how the use of these media (mass media then and social media today) serves to gratify the underlying needs of the audience that uses them. The popularity of this theory is such that in recent years, it has been used to address numerous studies on the use of social network services in all types of contexts and organizations [7–11].

1.1. Social Network Services in Organizations

The potential of these platforms as a means of gratification for their users has captured the attention of organizations of various kinds. Since social networks began to become popular in the early 2000s, many organizations have used these technologies to gratify the needs of their audiences.

However, social network services not only help organizations gratify the needs of their stakeholders, these platforms have also become alternative interaction tools for official websites, an economic means to create user communities around the organization, and, in many cases, an instrument to enhance the brand image of the institution [12].

In this sense, business organizations, on the one hand, and university organizations, on the other, are two of the entities in which social network services have gained the most traction. In both cases, the main objective of the organization is to convey information to their audiences in an agile way through a channel that facilitates dialogue between parties [13]. Therefore, we can say that both companies and universities use these platforms for communication purposes. However, the differential nuance is that, whereas university organizations adhere exclusively to this communicational purpose, business organizations also seek a transactional goal. That is, in the business field, these technologies also pursue the formalization of transactions, whether they are understood as customer acquisition or as selling products or services [14].

Thus, in recent years, platforms, such as Twitter, Facebook, and Instagram, have been integrated into the strategies of many organizations, until becoming, in many cases, the cornerstone of the actions carried out in their communication and marketing departments.

The literature specialized on the topic of the use of social network services within organizations, both in the company and in the university, includes numerous references. Table 1 lists a sample of some of the studies conducted in the last five years.

Regarding the use of social media in the business context, several works can be highlighted [15–19]. Balan's research [15] explored the way in which the topics of the Instagram posts of a major sports equipment brand influenced the recognition received by its publications. Their study revealed significant differences in views, comments, and likes received depending on the topic of publication.

Matosas López [16] analyzed the aspects that condition the propagation of the content of companies in the food sector on Twitter. The author examined the way in which the interactivity of the content (links or mentions), the vividness of the publications (photos or hashtags), the sentiment of the emoticons, and the posting time influenced the dissemination of messages.

Carlson et al. [17] studied the design characteristics of a sample of company pages on the social network Facebook. In this work, the authors observed that the design of the fan page determined the way the client perceived the organization, as well as the client's predisposition to build links with it.

Table 1. Studies on the use of social network services in organizations.

Author/s	Year	Platforms Considered	Type of Organization	Purpose of the Study
Laudano et al.	2016	Twitter	University	Dissemination of information about libraries collections and services
López-Pérez and Olvera-Lobo	2016	Facebook and Twitter	University	Distribution of research results
Cabrera Espín and Camarero	2016	Facebook	University	Analysis and comparison of digital communication channels
Kimmons et al.	2017	Twitter	University	Dialogic functionality of the platform
Balan	2017	Instagram	Business	Recognition received by the message depending on the publication topic
Matosas López	2018	Twitter	Business	Content sharing and propagation
Carlson et al.	2018	Facebook	Business	Client perception of the organization
Quitana Pujalte et al.	2018	Twitter	University	Use of corporate accounts in situations of reputational crisis
Wu et al.	2019	Facebook	University	Recognition obtained depending on the publication source
Mukherjee and Banerjee	2019	Facebook	Business	Impact that advertising insertions have on the user
Giakoumaki and Krepapa	2019	Instagram	Business	Influence of publication's tone on the volume of comments
Majumdar and Bose	2019	Twitter	Business	Associations between platform's activity and company market value

The research of Mukherjee and Banerjee [18], based on surveys, analyzed the impact that advertising insertions on Facebook had on the users of the platform. The authors showed that advertising can lead the audience to have a positive attitude towards the brand, also increasing the purchase intention of the products or services of the company.

Giakoumaki and Krepapa [19] analyzed how the contents of luxury brands on the Instagram platform can obtain greater or lesser recognition, depending on whether the publication came from one source or another. The authors found that the recognition when the source of the publication was a personal account was greater than when the content was published by an influencer or by the corporate account of the company.

Finally, Majumdar and Bose [20], applying a multi-period analysis, studied, in a sample of manufacturing firms, the relationship between Twitter related activities and the company market value. The researches revealed the existence of positive associations between the distribution of product-related information in this social network and the firm's value.

Among studies that take university organizations as an object of analysis, several works stand out [21–25]. Laudano et al. [21] examined the Twitter presence of a sample of university libraries. Their findings revealed that, although libraries use this platform to disseminate information about collections, services, or the promotion of activities, its use is in general diffuse and poorly planned.

López-Pérez and Olvera-Lobo [24] explored the use of social media technologies for the distribution of research results in public university organizations. The authors confirmed that approximately 40% of the institutions examined used their corporate accounts on Facebook and Twitter to disseminate this type of content.

Cabrera Espín and Camarero [22] analyzed the different communication channels used by a sample of university institutions. Among other results, the researchers addressed that approximately 80% of the students turned to the university Facebook account to learn about the current affairs of their school, even more than on the school's own website.

Kimmons et al. [26], using a wide sample of publications, investigated the institutional uses of Twitter in colleges and universities. Their study suggested that even though these technologies are commonly considered as dialogic platforms, their use, in many cases, remains remarkably monologic, focusing all attention on the unidirectional distribution of information of an institutional nature.

Quitana Pujalte et al. [23] examined the ways universities use their corporate accounts to respond to situations of reputational crisis. The study showed that the university's Twitter profile can be used, in such circumstances, to redirect traffic to the institutional website or to official press releases.

Finally, Wu et al. [25] analyzed the comments that the publications of a sample of universities are capable of generating on Facebook. The authors noted that publications that use a friendly and familiar tone receive a greater volume of comments than those that use a more direct and authoritative tone.

1.2. The Efficiency of Social Network Services Management in Organizations

As we can see, both business and university organizations use these technologies regularly and for different purposes. However, the keys to be considered by these organizations for developing efficient management of their platforms continue to be debated. Some authors hold that one of the problems in the management of these technologies lies in the lack of professionalization of the work teams [27]. Others point out that the management of social network services in organizations suffers from a lack of strategic planning [21].

The deficiencies in social media management are evident, but academics and professionals do maintain a firm consensus on which indicator to use to evaluate whether this management is adequate. This indicator is the recognition that the audience of the account gives to the publications of the account when they see their needs gratified.

As soon as the user perceives that the need that had originated his or her connection with the organization has been satisfied, he or she reacts positively by resorting to the relevant functionalities enabled in the platform. According to some authors [9,16,28], the user manifests this recognition of the organization by sharing its content or marking it as a favorite.

Even though the efficiency of management of social network services seems to have as an unquestionable indicator of success the recognition of content, either in the form of sharing or favoriting publications, the way to maximize this indicator is still under research. Fortunately, the enormous volume of information hosted in social network services enables a detailed study of the activity and behavior of its users.

1.3. Objectives

The millions of interactions that occur daily between organizations and users in these platforms generate millions of terabytes of information. The application of machine learning algorithms and multiple linear regressions allows us to extract the underlying knowledge in these immense information banks. Nevertheless, the ultimate goal of these techniques is the identification of trends, patterns, or models that facilitate decision-making and allow the organization to manage these technologies [29,30] efficiently.

Although some works have previously applied machine learning algorithms and multiple linear regressions to examine the activity occurred on social media, the dynamic and changing nature of these spaces requires constant updates of this knowledge [2]. An in-depth analysis of the trends and patterns of use is, without a doubt, the basis on which professionals in this field develop efficient management of these technologies in their organizations.

This research, based on the application of machine learning algorithms and multiple linear regressions, aims to provide information that serves to update this knowledge about the media. Consequently, the main objective of this work is to identify the variables that allow organizations to manage their social network services efficiently.

The object of study is the official Twitter accounts of university organizations in Spain. The social network service Twitter is taken as the object of research for the ease of access to the data. Likewise, the decision to opt for university organizations is due to the purely communicative purpose of these organizations, leaving aside the transactional objective of business organizations. Finally, the selection of Spanish institutions is justified both by the huge social media activity shown by universities in this country and by the variety of publication topics traditionally addressed by their accounts [31].

In this setting, the research analyzes how certain characteristics of the content, on the one hand, and the message of the content, on the other hand, increase the recognition of publications through retweets and favorites.

The characteristics of the content considered are publication volumes, publication components, and publication moments, whereas the effect of the message of the content focuses on the publication topics. This research will, therefore, answer the following two research questions:

Research Question I (RQI): What are the publication volumes, publication components, and publication moments that increase content recognition in the form of retweets and favorites?

Research Question II (RQII): What are the publication topics that increase content recognition in the form of retweets and favorites?

2. Materials and Methods

Applying the postulates of studies on the analysis of social network services, and following the recommendations of Saura et al. [32], this study was organized into three stages: (1) sample design and data extraction, (2) data cleaning and organization, and (3) data analysis. These three stages (see Figure 1) are described below.

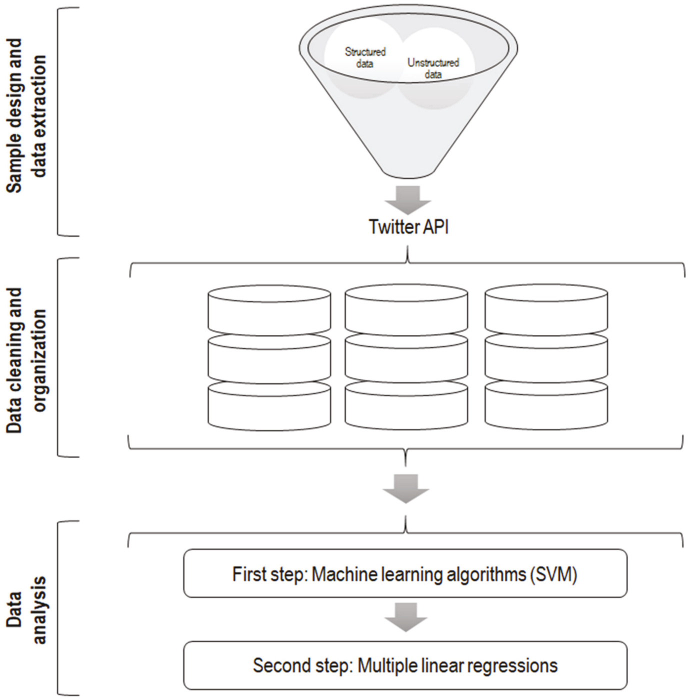

Sample design and data extraction
Structured data
Unstructured data
Twitter API
Data cleaning and organization
Data analysis
First step: Machine learning algorithms (SVM)
Second step: Multiple linear regressions

Figure 1. Stages of the methodology.

2.1. Sample Design and Data Extraction

The researchers used a sample of Spanish university organizations. The selection of sampling elements was based on two of the most recognized rankings for assessing the activity of university institutions: the Webometrics list [33] and the Academic Ranking of World Universities (ARWU), also known as Shanghai ranking [34].

The authors took as their starting point the institutions located in the first fifteen positions of the Webometrics ranking in Spain in 2019 to then check whether these organizations appeared among the global top 500 of the ARWU of that same year. The authors selected only those institutions that rank in the top fifteen in Spain on the Webometrics list and, at the same time, among the top 500 in the world, according to the ARWU. This screening reduced the sample to ten organizations. The institutions were the University of Barcelona, Complutense University of Madrid, Autonomous University of Barcelona, University of Valencia, University of Granada, Autonomous University of Madrid, Polytechnic University of Catalonia, Polytechnic University of Valencia, Polytechnic University of Madrid and Pompeu Fabra University.

Once the sample was selected, the researchers extracted from the Twitter platform, all the content published by the official accounts of the ten organizations over a one-year period. Following the procedure of previous studies [23,35], the data were extracted through Twitter's API using the service provider Twitonomy. This process led to the gathering of 21,771 publications, in addition to the recognition obtained by each of them in terms of retweets and favorites.

2.2. Data Cleaning and Organization

The compiled data set was stored for cleaning and organization, extracting a total of thirty-nine variables arranged into six categories: (a) Publication volumes, (b) Publication components, (c) Publication day of the week, (d) Publication time slot, (e) Publication topic, and (f) Recognition obtained by the publication (see Table 2). These six categories, and the variables contained in them, were determined in accordance with previous research.

Table 2. Variables extracted from the data set.

Category	No. of Variables	Name of the Variables
(a) Publication volumes	3	Original Tweets, Retweets, and Replies
(b) Publication components	3	Links, Mentions, and Hashtags
(c) Publication day of the week	7	Pub. On Mon., Pub. On Tue., Pub. On Wed., Pub. On Thu, Pub. On Fri., Pub. On Sat., Pub. On Sun.
(d) Publication time slot	8	Pub. 8:00 to 10:00, Pub. 11:00 to 13:00, Pub. 14:00 to 16:00, Pub. 17:00 to 19:00, Pub. 20:00 to 22:00, Pub. 23:00 to 1:00, Pub. 2:00 to 4:00, Pub. 5:00 to 7:00
(e) Publication topic	16	Central topic discussed in the pub.
(f) Recognition obtained by the publication	2	Retweeted Pubs. and Favorite Pubs.
Total	39	

The variables gathered in categories (a), (b), (c), (d), and (e) were taken as independent variables, whereas the variables in category (f) were used as dependent variables.

Publications volumes were defined considering the proposal of Bruns and Stieglitz [36]. Publication components were operationalized through the adaptation of post characteristics from De Vries et al. [37]. Publication moment, covering the categories of publication day and publication time slot was based on the analysis of Valerio Ureña et al. [38] in their study on associations between the moment of publication in social media and the engagement concept. Publication topics were addressed in accordance with the proposal of García [39] in her study on communication management in social networks services. And finally, the category that represented the recognition obtained by the publication was determined following the recommendations of authors such as Chen [9] or Pletikosa Cvijikj and Michahelles [28], among others.

The independent variables were clearly separated and differentiated from each other. For instance, a publication could be "Original Tweet", "Retweet", or "Reply", but never "Original Tweet" and "Reply" at the same time. Similarly, a message with a unique publication ID can only be posted on a specific day of the week. In the same way, a publication can not be categorized in two time slots at the same time.

Nevertheless, there could be potential correlations between variables placed in different categories. Thus, for example, publication days or publication time slots could be correlated with the topics of publication. This could lead us to think that the variables in the publication topics´ category could also be considered as independent variables. However, this work, in line with previous studies on the efficiency of social media management in organizations, used as independent variables those commonly taken by the research community when evaluating the recognition of publications [9,16,28]. That is, the variables contained in the category (f), Retweeted Pubs. and Favorite Pubs.

2.3. Data Analysis

To carry out the data analysis, the authors used a two steps approach. First, machine learning algorithms were applied for the classification of publication topics (Category e). Second, multiple linear regressions were used to reveal the volumes (Category a), components (Category b), publication moments (Categories c and d), and publication topics (Category e) that increased the recognition of content.

2.3.1. First Step: Machine Learning Algorithms

The authors applied machine learning algorithms to classify the publication topics (Category e). These publication topics would be used as independent variables in the multiple linear regression carried out in the second step of the data analysis.

In the field of social network services, machine learning algorithms are used to conduct categorizations or classifications of text publications [40]. These systems allow organizations to classify thousands or millions of pieces of text efficiently, and comfortably, for later exploration.

The textual information analyzed using machine learning algorithms is classified as unstructured information. These data do not adhere to a previously defined scheme; therefore, their processing requires the application of certain rules (idiomatic, grammatical, and semantic) to extract the information they contain.

Specifically for the platform under study, the methodologies based on Twitter Analytics approaches addressed by Goonetilleke et al. [41], Kumar et al. [42] or Lin and Ryaboy [43] generally use machine learning algorithms, either to analyze the sentiment of publications or to study specific hashtags. Examples of works focused on the analysis of the sentiment of publications (positive, negative, and neutral) are the studies by Hoeber et al. [44] or Saura et al. [32]. Whereas, examples of investigations focused on the observation of specific hashtags are the works of Lakhiwal and Kar [45] or De Maio et al. [46].

With respect to the techniques used in these methods, the following stood out: decision trees (DT), random forest (RF), Naïve Bayes classifier (NBC), logistic regression (LR), k-nearest neighbors (kNN) and support vector machines (SVM) [47–49]. However, whereas many of these techniques can be effective in determining the sentiment of publications or for hashtag examinations, the most appropriate technique for the classification of complex publication topics, and the one that offers the highest accuracy, is the SVM technique [14].

The SVM technique applied in the present study used, specifically, the linear Kernel function as a classification method. This general Kernel function is defined as follows:

$$\mathrm{K}\ (\mathrm{x}_i, \mathrm{x}_j) = \Phi\ (\mathrm{x}_i)\cdot\Phi\ (\mathrm{x}_j) \tag{1}$$

where K (x*i*, x*j*) is the core function, and Φ (x*i*) represents the mapping space associated with the vectors.

The machine learning algorithm used for the classification of publication topics (Category e) was a supervised machine learning algorithm. With supervised machine learning algorithms, there exists an initial set of already labeled data with input-output pairs that allows for training of the predictive model. From this initial data set, the algorithm learns to assign the appropriate output label to each incoming element in the model [50]. In the case of the specific application of these algorithms in the classification of texts in social media, the labeled data set is typically created, on a small scale, via the intervention of a subject or group of subjects who assign each publication the most appropriate label in each case.

In line with previous research, the classification was performed through the text classification API of the MonkeyLearn library [51,52]. This text classification API uses the JASON notation protocol in JavaScript, also allowing the researcher to carry out, before classification, a training process with the algorithm.

After carrying out this training process, manually categorizing 300 publications, the algorithm had the necessary knowledge to develop a personalized machine learning model. This model allowed the classification of the texts of each of the 21,771 publications into one publication topic.

2.3.2. Second Step: Multiple Linear Regressions

The researchers used multiple linear regressions to discover the publication volumes (Category a), publication components (Category b), publication moments (Categories c and d), and publication topics (Category e) that increased the recognition of content.

In the context of social network services, multiple linear regressions focus on quantitative analyses of activity metrics from organizations and users [16].

The information, in the form of metrics that is analyzed by applying multiple linear regressions is generally regarded as structured information. These data are collected in predefined fields and presented using tables of values in which fields and cases are represented in columns and rows, respectively.

The analyses of activity metrics on these platforms can be carried out using techniques such as simple linear regressions (SLR), structural equation modeling (SEM), or even descriptive explorations. Examples of studies that use these techniques are the works of Valerio Ureña and Serna Valdivia [53], Pletikosa Cvijikj and Michahelles [28], or Alonso [54], among others. However, although these techniques are widely accepted, multiple linear regression is probably the most effective technique when the purpose is knowing not only the influence of the independent variables on the dependent ones individually, but also the joint potential of these within the predictive model [16].

The general equation, which is used to represent the multiple linear regression, is expressed as:

$$Y_i = \alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_k X_{ik} + \varepsilon_i \tag{2}$$

where α is the constant term of the model, Yi is the dependent variable, Xi represents the independent variables, β represents the regression coefficients, and εi is the error or average of residuals.

The multiple linear regression allowed us to reveal the volumes, components, publication moments, and publication topics that increased the recognition of content. In this analysis, the authors took all the thirty-nine variables considered in the study. Thirty-seven acted as independent variables and two as dependent variables. These thirty-seven independent variables corresponded to the categories (a) Publication volumes, (b) Publication components, (c) Publication day of the week, (d) Publication time slot, and (e) Publication topic. The two dependent variables were those corresponding to category (f) Recognition obtained by the publication (Retweeted Pubs. and Favorite Pubs.).

3. Results

3.1. First Step: Machine Learning Algorithms

The SVM technique applied, using the linear Kernel function as a classification method, reflected the existence of sixteen publication topics: General news, Scholarships, Science and technology, Contests, Culture and exhibitions, Sports, Entrepreneurship, Complementary training, Gender equality, Institutional information, Employability, Research, Seminars and conferences, Awards and recognitions, Health and green environment, and Volunteering.

These publication topics were determined by the researchers and validated by a panel of five judges who were experts on the management of social networks services in different organizational contexts.

In line with previous research, the authors used the Krippendorff's alpha to measure the accuracy of the text classification carried out by the supervised machine learning algorithm [32]. The Krippendorff's alpha value obtained (0.886), which is above the recommended threshold of 0.800, indicated that the supervised machine learning algorithm had been properly trained, and its predictive power was accurate enough.

The descriptive exploration of the publication topics addresses the presence of differences in the recognition obtained by the different topics. Table 3 reveals that there were differences in the way in which the content of each publication topic was recognized and what topics obtained greater recognition.

Table 3. Retweets and favorites obtained by publication topic.

Publication Topic	Number of Tweets	% of Total Tweets	Average Number of Retweets Obtained Per Pub.	Average Number of Favorites Obtained Per Pub.
General news	2458	11.29%	18.87	9.21
Scholarships	429	1.97%	11.21	11.16
Science and technology	1960	9.00%	13.68	8.37
Contests	267	1.23%	9.06	6.41
Culture and exhibitions	1437	6.60%	10.99	6.23
Sports	814	3.74%	11.87	5.01
Entrepreneurship	308	1.41%	7.97	5.25
Complementary training	1228	5.64%	8.01	4.18
Gender equality	700	3.22%	50.98	26.14
Institutional information	4382	20.13%	13.87	7.21
Employability	685	3.15%	8.12	4.18
Research	1796	8.25%	13.08	7.38
Seminars and conferences	1968	9.04%	7.98	4.07
Awards and recognitions	1907	8.76%	11.94	3.98
Health and green environment	1177	5.41%	18.54	11.99
Volunteering	255	1.17%	13.87	5.29
Total	21,771	100.00%	45.84	24.98

This descriptive examination revealed that institutional information and general news were the most recurring topics, accounting for 20.13% and 11.29% of the publications, respectively.

The average number of retweets and favorites received per publication showed that the contents that achieved the greatest recognition were those related to the gender equality topic. Paradoxically, this topic, with the highest retweet and favorite average, represented only 3.22% of all publications. Therefore, it seems to be clear that certain topics get far more recognition than others.

3.2. Second Step: Multiple Linear Regressions

Two multiple linear regression were performed, one for the dependent variable Retweeted Pubs. and another for the dependent variable Favorite Pubs. The results obtained from these analyses allowed the identification of the variables that increased content recognition in the form of retweets and favorites within the respective models. To examine the explanatory power of each independent variable, the items of the categories (a), (b), (c), (d), and (e) were introduced in their respective model as individual indicators. The researchers applied here the stepwise method for incorporating the variables.

In the first regression, the one performed for the Retweeted Pubs., the item "Links" ($\beta = 0.560$, p-value < 0.0001), was added in the first step of the procedure. The variable "Hashtags" ($\beta = 0.455$, p-value < 0.005) was introduced in the second step. Finally, the item "Gender equality" ($\beta = 0.447$, p-value < 0.0001) was added in the third step.

The model for this first dependent variable (Retweeted Pubs.) was significant as a whole ($F = 78.341$, p-value < 0.0001), optimally explaining the variance of the dependent variable with values of $R = 0.976$ and $R^2 = 0.951$. Therefore, this first regression showed the impact of the variables "Links", "Hashtags", and "Gender equality" when predicting the recognition of content published through retweets (see Table 4).

Table 4. Coefficients of multiple linear regressions.

	(f) Recognition Obtained by the Publication			
	Retweeted Pubs.		Favorite Pubs.	
Category/Variable	**B**	***p*-Value**	**B**	***p*-Value**
(a) Publication volumes				
Original Tweets	−0.164	0.281	0.198	0.003 *
Retweets	−0.098	0.326	0.014	0.624
Responses	0.011	0.908	0.077	0.346
(b) Publication components				
Links	0.560	0.000 **	−0.017	0.899
Mentions	−0.048	0.680	0.112	0.016
Hashtags	0.455	0.001 *	0.090	0.097
(c) Publication day of the week				
Pub. On Monday	−0.111	0.538	−0.421	0.074
Pub. On Tuesday	−0.121	0.435	−0.094	0.698
Pub. On Wednesday	−0.107	0.441	−0.127	0.518
Pub. On Thursday	−0.091	0.437	0.228	0.374
Pub. On Friday	−0.157	0.381	0.124	0.493
Pub. On Saturday	−0.153	0.207	0.049	0.409
Pub. On Sunday	−0.006	0.981	−0.054	0.364
(d) Publication time slot				
Pub. 8:00 to 10:00	−0.071	0.781	0.237	0.004 *
Pub. 11:00 to 13:00	−0.088	0.601	0.184	0.081
Pub. 14:00 to 16:00	−0.009	0.971	0.091	0.172
Pub. 17:00 to 19:00	−0.016	0.902	−0.017	0.791
Pub. 20:00 to 22:00	0.131	0.547	−0.039	0.514
Pub. 23:00 to 1:00	0.059	0.611	0.009	0.843
Pub. 2:00 to 4:00	0.069	0.654	−0.062	0.185
Pub. 5:00 to 7:00	−0.157	0.135	−0.124	0.018
(e) Publication topic				
General news	0.185	0.420	0.021	0.734
Scholarships	−0.014	0.750	0.180	0.121
Science and technology	−0.039	0.537	0.074	0.117
Contests	0.517	0.427	0.092	0.092
Culture and exhibitions	0.066	0.905	0.071	0.151
Sports	−0.087	0.547	0.263	0.341
Entrepreneurship	−0.124	0.411	0.181	0.512
Complementary training	−0.159	0.195	0.025	0.663
Gender equality	0.447	0.000 **	0.531	0.001 *
Institutional information	0.039	0.732	0.018	0.903
Employability	−0.058	0.701	0.209	0.214
Research	−0.109	0.381	−25.852	0.468
Seminars and conferences	0.091	0.514	0.034	0.584
Awards and recognitions	−0.113	0.145	0.019	0.607
Health and green environment	0.127	0.584	−0.037	0.484
Volunteering	0.052	0.552	0.017	0.803

* *p*-value < 0.005; ** *p*-value < 0.0001.

For the second regression, the one developed for the variable Favorite Pubs., the item "Original Tweets" (β = 0.198, *p*-value < 0.005) appeared in the first step of the process. The variable added in the second step was called "Pub. 8:00 to 10:00" (β = 0.237, *p*-value < 0.005). To finish, the item "Gender equality" (β = 0.531, *p*-value < 0.005) appeared in the third step.

The model for the second dependent variable (Favorite Pubs.) was also significant (F = 311.278, *p*-value < 0.0001), adequately explaining the variance of this variable with values of R = 0.931 and

$R^2 = 0.917$, respectively. Therefore, this second regression revealed the influence of the variables "Original Tweets", "Pub. 8:00 to 10:00", and "Gender equality" on the recognition obtained, through favorites, of the content published by the organization (see Table 4).

To corroborate the validity of the above regressions, the authors analyzed the residuals of both using the Shapiro–Wilk test and the Durbin–Watson test.

The Shapiro–Wilk test was performed to see whether the values of the standardized residuals followed a normal distribution. The *p*-values above 0.050 in the two regressions (0.851 for the first and 0.721 for the second) confirmed that the residuals were normally distributed [55]. The Durbin–Watson test served to verify whether the assumption of independence of residuals was met. The values of this indicator between 1 and 3 in both regressions (1.847 for the first and 1.425 for the second) verified that the requirement of independence of residual was satisfied [56]. The values in the Shapiro–Wilk and Durbin–Watson tests confirmed that the predictive models obtained from the multiple linear regressions carried out were adequate and robust.

4. Discussion

Different authors have highlighted the need for organizations to invest in efficient management in social network services. Some studies suggest that these platforms require professionalized management systems and that their management cannot be left to nonspecialized profiles [12,27]. Other authors claim that the way many organizations handle these technologies lacks strategic vision [21]. Along the same lines, there are studies that indicate that organizations should not settle for using their accounts to build their institutional image but rather must also protect the reputation of the organization [57]. Some authors claim that properly managed, social network services can even serve as a customer acquisition tool [58].

The findings of this study provide academics and professionals with the necessary knowledge to efficiently manage their use of these technologies, enabling organizations to satisfy many of the aforementioned purposes. The results obtained, thanks to the application of machine learning algorithms and multiple linear regressions, allow us to answer the two research questions posed: the one that concerns the characteristics of the content (publication volumes, publication components, and publication moments) and the one related to the message of the content (publication topics).

4.1. Volumes, Components, and Publication Moments That Increase Content Recognition (RQ1)

The multiple linear regressions showed that content recognition through retweets was conditioned on the use of links and hashtags in publications, whereas recognition by favorites was fundamentally determined by the frequency of original tweets and a publication time between 8:00 and 10:00 a.m. The influence of these four variables had a positive valence. Thus, greater exploitation of links, hashtags, original tweets, and early-morning publication boosts the recognition achieved by the organization in the form of retweets and favorites.

Such results corroborate, for example, the findings of Túñez López et al. [34] in their work on the use of Facebook and Twitter as communication channels. Those authors highlighted the value of links as an essential element in any message.

Regarding the use of hashtags, the results are in line with those of Guzmán Duque et al. [58] in their study on the impact of the use of Twitter in the organizational field. In this work, the authors highlighted the potential of these markers in facilitating the promotion and projection of the organization to the audience.

As for the frequency of original tweets, the findings of this work corroborate what was indicated by Chen [9] in a study on uses and gratifications on Twitter: a high publication frequency of original content acts as a motivating factor that encourages the subject to interact with other users.

Finally, with respect to the publication moment, the results are aligned with the findings of Hanifawati et al. [59] in their work on the management of corporate Facebook accounts. In that study, the researchers emphasized that the messages in the most active time slots, those in which the user is more likely to visit the platform, increase both the amount of shared content and the comments received on it.

4.2. Publication Topics That Increase Content Recognition (RQII)

The multiple linear regressions demonstrated that content recognition through retweets and favorites can be influenced using topics related to gender equality. Therefore, the use of publications with this thematic approach increases the recognition achieved by the organization in the form of retweets and favorites.

Although it is true that other authors have highlighted the importance of the theme of the publication in the context of social network services in organizations [27,38], there are few studies that examine this issue in depth. When this has been done, the analyses tend to focus more on the sentiment or tone of the publications [25,60], or on superficial explorations of hashtags or predetermined search terms [34,61]. Consequently, these studies generally ignore most of the text of the publication and the semantics of the expressions contained in it. The present work, thanks to the use of a supervised machine learning algorithm, previously trained by the researchers, allowed for a highly adjusted text classification of the publications. This text classification, carried out in the first step of the analysis, allowed us to identify the publication topics that were used later in the multiple linear regression performed in the second step of the data analysis.

The analysis carried out by the authors revealed differences in the recognition obtained by the different publication topics. These findings are in line with the findings of Pletikosa Cvijikj and Michahelles [28] in their study on engagement factors in online communities within Facebook. Those authors pointed out that the type of content published by the organization can indeed determine the recognition obtained in its audience.

The findings achieved in the present research revealed that, paradoxically, topics with a smaller weight over the total number of publications, such as those that address topics related to gender equality, were the most successful in terms of recognition by the audience.

5. Conclusions

Although in recent years some works have used machine learning algorithms and multiple linear regressions to examine the activity that has occurred on social media, these studies tend to focus exclusively on content characteristics (publication volumes, publication components, and publication moments) [16,62] or in its message (tone, sentiment, or publication topic) [60,63]. However, few studies have examined both topics simultaneously.

Perhaps the most emblematic of these studies is that of Pletikosa Cvijikj and Michahelles [28]. Their work, like the present one, considered the characteristics of the content (publication volumes, publication components, and publication moments) and the message of the content (publication topics). Nevertheless, their study analyzed a data set smaller than that of the present study and applied a text classification with only three publication topics (information, entertainment, and rewards), as opposed to the sixteen considered here.

The present research not only combines a study of the characteristics (volumes, components, and moments) and the message (topics) of the publications, it also addresses this challenge more comprehensively than previous works. In the authors' opinion, the examination of the characteristics and the message of the publications, in addition to the two steps analysis approach applied in the investigation, are among the key values of the current study. The supervised machine learning algorithm applied in the first term allowed the classification of the texts into the publication topics. Knowing publication topics besides publication volumes, publication components, and publication moments, the authors applied multiple linear regressions to discover the influence of all these variables on the recognition of content.

The results gathered to answer the first research question (RQI) are in line with the findings of previous studies, whereas the results obtained for the second question (RQII) provide novel and relevant information on the current field of investigation.

To this second point, regarding the publication topic, the findings confirmed that publications on topics of gender equality achieve much higher recognition than those obtained by content focused on other topics. In the opinion of the authors, this situation is conditioned by the recent social sensitivity around this issue. Likewise, given that no organization goes uninfluenced by social issues of this nature, the knowledge derived from these results in the context of university organizations is likely to be extended to the field of business organizations.

On the other hand, this finding can prompt reflections on the importance of media professionals' managing these technologies adequately and being able to identify, at all times, the topics of interest to their audience, adapting the content of their organizations to these preferences.

In view of all the above, the authors confirm the value of applying machine learning algorithms and multiple linear regressions to carry out an in-depth analysis of the enormous amount of information generated by social network services gaining new knowledge about trends and usage patterns in the media. This renewal will ultimately be the basis for the efficient management of social network services in the organizational field.

6. Limitations and Further Research

This paper also suffers from several limitations. The sample, although significant, could be amplified to examine more deeply the observed phenomena.

In addition, future research could also consider complementing the analyzes carried out in the present study with other analytical approaches. A work like the present one could be complemented, for example, by using network centrality analysis [42] or OLAP (On-Line Analytical Processing) techniques [64,65].

Centrality analysis, generally supported in JUNG (Java Universal Network-Graph) open source frameworks, are used to identify who is the most important user in the network; revealing in a graphical way who gets more retweets (Degree Centrality), which is the most influential user (Eigenvector Centrality), or the number of shortest paths in which the user distributes the information (Betweenness Centrality).

Likewise, OLAP techniques allow the extraction of information related to user behaviors, emerging topics, or trends, providing generic multidimensional models for the analysis of data on social network services.

The aforementioned issues address new avenues for research in this field, confirming that further investigation is still needed to expand our understanding of the activity on social media.

Author Contributions: Conceptualization, methodology, formal analysis, investigation, data curation, writing—original draft, project administration, L.M.-L.; validation, writing—review and editing, visualization, supervision, A.R.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hootsuite & We Are Social; Kemp, S. Digital 2020: Global Digital Overview; Singapore. Available online: https://datareportal.com/reports/digital-2020-singapore?rq=singapore (accessed on 1 May 2020).
2. Gómez-García, M.; Matosas-López, L.; Palmero-Ruiz, J. Social Networks Use Patterns among University Youth: The Validity and Reliability of an Updated Measurement Instrument. *Sustainability* **2020**, *12*, 3503. [CrossRef]
3. Sandoval Romero, Y.; Aguaded Gómez, J.I. Nuevas audiencias, nuevas responsabilidades. La competencia mediática en la era de la convergencia digital. *ICONO14* **2012**, *10*, 8–22. [CrossRef]

4. Sánchez Carrero, J.; Contreras Pulido, P. De cara al prosumidor: Producción y consumo empoderando a la ciudadanía 3.0. *ICONO14* **2012**, *10*, 62–84. [CrossRef]
5. Katz, E.; Blumler, J.G.; Gurevitch, M. Uses and Gratifications Research. *Public Opin. Q.* **1974**, *37*, 509–523. [CrossRef]
6. Ruggiero, T.E. Uses and Gratifications Theory in the 21st Century. *Mass Commun. Soc.* **2000**, *3*, 3–37. [CrossRef]
7. Raacke, J.; Bonds-Raacke, J. MySpace and Facebook: Applying the Uses and Gratifications Theory to Exploring Friend-Networking Sites. *Cyberpsychol. Behav.* **2008**, *11*, 169–174. [CrossRef]
8. Smock, A.D.; Ellison, N.B.; Lampe, C.; Wohn, D.Y. Facebook as a toolkit: A uses and gratification approach to unbundling feature use. *Comput. Hum. Behav.* **2011**, *27*, 2322–2329. [CrossRef]
9. Chen, G.M. Tweet this: A uses and gratifications perspective on how active Twitter use gratifies a need to connect with others. *Comput. Hum. Behav.* **2011**, *27*, 755–762. [CrossRef]
10. Tarullo, R. ¿Por qué los y las jóvenes están en las redes sociales? Un análisis de sus motivaciones a partir de la teoría de usos y gratificaciones. *Prism. Soc.* **2020**, *29*, 222–239.
11. Cuevas-Molano, E.; Sánchez Cid, M.; Matosas-López, L. Análisis bibliométrico de estudios sobre la estrategia de contenidos de marca en los medios sociales. *Comun. Soc.* **2019**, *2019*, 1–25. [CrossRef]
12. Casanoves Boix, J.; Küster Boluda, I.; Vila López, N. ¿Por qué las instituciones de educación superior deben apostar por la marca? *Rev. Investig. Educ.* **2018**, *37*, 111–127. [CrossRef]
13. Oviedo-García, M.A.; Muñoz-Expósito, M.; Castellanos-Verdugo, M.; Sancho-Mejías, M. Metric proposal for customer engagement in Facebook. *J. Res. Interact. Mark.* **2014**, *8*, 327–344. [CrossRef]
14. Matosas-López, L. Cómo distintos tipos de organización gestionan su presencia en plataformas sociales. In Proceedings of the XX International Conference on Knowledge, Culture, and Change in Organizations, Chicago, IL, USA, 20–21 October 2020.
15. Balan, C. Nike on Instagram: Themes of branded content and their engagement power. In Proceedings of the CBU International Conference, Prague, Czech Republic, 22–24 March 2017; Volume 5, pp. 13–18.
16. Matosas López, L.M. Variables of twitter´s brand activity that influence audience spreading behavior of branded content. *ESIC Mark. Econ. Bus. J.* **2018**, *44*, 525–546. [CrossRef]
17. Carlson, J.; Rahman, M.; Voola, R.; De Vries, N. Customer engagement behaviours in social media: Capturing innovation opportunities. *J. Serv. Mark.* **2018**, *32*, 83–94. [CrossRef]
18. Mukherjee, K.; Banerjee, N. Social networking sites and customers' attitude towards advertisements. *J. Res. Interact. Mark.* **2019**, *13*, 477–491. [CrossRef]
19. Giakoumaki, C.; Krepapa, A. Brand engagement in self-concept and consumer engagement in social media: The role of the source. *Psychol. Mark.* **2019**, *37*, 457–465. [CrossRef]
20. Majumdar, A.; Bose, I. Do tweets create value? A multi-period analysis of Twitter use and content of tweets for manufacturing firms. *Int. J. Prod. Econ.* **2019**, *216*, 1–11. [CrossRef]
21. Laudano, C.N.; Planas, J.; Kessler, M.I. Aproximacones a los usos de twitter en bibliotecas universitarias de Argentina. *An. Doc.* **2016**, 19. [CrossRef]
22. Cabrera Espín, S.I.; Camarero, E. Comunicación de la ciencia y la tecnología en las universidades ecuatorianas: Estudio preliminar del impacto y percepción entre la población universitaria. *Rev. Comun. SEECI* **2016**, *40*, 27. [CrossRef]
23. Quitana Pujalte, L.; Sosa Valcarcel, A.; Castillo Esparcia, A. Acciones y estrategias de comunicación en plataformas digitales. El caso Cifuentes. *Prism. Soc.* **2018**, *22*, 247–270.
24. López-Pérez, L.; Olvera-Lobo, M.-D. Comunicación pública de la ciencia a través de la web 2.0. El caso de los centros de investigación y universidades públicas de España. *El profesional de la información* **2016**, *25*, 441. [CrossRef]
25. Wu, J.; Chen, J.; Chen, H.; Dou, W.; Shao, D. What to say on social media and how: Effects of communication style and function on online customer engagement in China. *J. Serv. Theory Pract.* **2019**, *29*, 691–707. [CrossRef]
26. Kimmons, R.; Veletsianos, G.; Woodward, S. Institutional Uses of Twitter in U.S. Higher Education. *Innov. High. Educ.* **2017**, *42*, 97–111. [CrossRef]
27. Laaser, W.; Brito, J.G.; Toloza, E.A. El uso de redes sociales por parte de las universidades a nivel institucional. Un estudio comparativo. *Red Rev. Educ. A Distancia* **2012**, *32*, 231–239.
28. Pletikosa Cvijikj, I.; Michahelles, F. Online engagement factors on Facebook brand pages. *Soc. Netw. Anal. Min.* **2013**, *3*, 843–861. [CrossRef]

29. Golchha, N. Big Data—The information revolution. *Int. J. Appl. Res.* **2015**, *1*, 791–794.
30. García-González, J.D.; Skrita, A. Predicting academic performance based on students' family environment: Evidence for Colombia using classification trees. *Psychol. Soc. Educ.* **2019**, *11*, 299–311. [CrossRef]
31. Matosas-López, L. Divergencias en el uso de redes sociales en universitarios de los grados de Economía y Marketing. In *Aproximación Periodística y Educomunicativa al Fenómeno de las Redes Sociales*; Sanchez, A.M.d.V.D.J.S., Ed.; McGraw Hill: Madrid, Spain, 2019; pp. 697–708.
32. Saura, J.R.; Herraez, B.R.; Reyes-Menendez, A. Comparing a traditional approach for financial brand communication analysis with a big data analytics technique. *IEEE Access* **2019**, *7*, 37100–37108. [CrossRef]
33. Marciniak, R. Propuesta metodológica para la aplicación del benchmarking internacional en la evaluación de la calidad de la educación superior virtual. *Rev. Univ. Soc. Del Conoc.* **2013**, *12*, 46–61. [CrossRef]
34. Túñez López, M.; Valdiviezo Abad, C.; Martínez Solana, Y. Las redes sociales en la gestión de la comunicación universitaria. *Opción* **2015**, *6*, 852–874.
35. Alkadri, M.F.; Istiani, N.F.F.; Yatmo, Y.A. Mapping Social Media Texts as the Basis of Place-Making Process. *Procedia Soc. Behav. Sci.* **2015**, *184*, 46–55. [CrossRef]
36. Bruns, A.; Stieglitz, S. Towards more systematic Twitter analysis: Metrics for tweeting activities. *Int. J. Soc. Res. Methodol.* **2013**, *16*, 91–108. [CrossRef]
37. De Vries, L.; Gensler, S.; Leeflang, P.S.H. Popularity of Brand Posts on Brand Fan Pages: An Investigation of the Effects of Social Media Marketing. *J. Interact. Mark.* **2012**, *26*, 83–91. [CrossRef]
38. Valerio Ureña, G.; Herrera-Murillo, D.J.; Rodríguez-Martínez, M.D.C. Asociación entre el momento de publicación en las redes sociales y el engagement: Estudio de las universidades Mexicanas. *Palabra Clave* **2014**, *17*, 749–772. [CrossRef]
39. García García, M. Universidad y medios sociales. Gestión de la comunicación en la universidad española. *Prism. Soc.* **2018**, *22*, 21–36.
40. Thai, M.T.; Wu, W.; Xiong, H. *Big Data in Complex and Social Networks*; Chapman & Hall/CRC Press: Boca Raton, FL, USA, 2016.
41. Goonetilleke, O.; Sellis, T.; Zhang, X.; Sathe, S. Twitter analytics: A big data management perspective. *ACM Sigkdd Explor. Newsl.* **2014**, *16*, 11–20. [CrossRef]
42. Kumar, S.; Morstatter, F.; Liu, H. *Twitter Data Analytics*; Springer: New York, NY, USA, 2013.
43. Lin, J.; Ryaboy, D. Scaling Big Data Mining Infrastructure: The Twitter Experience. *Sigkdd Explor.* **2012**, *14*, 6–19. [CrossRef]
44. Hoeber, O.; Hoeber, L.; El Meseery, M.; Odoh, K.; Gopi, R. Visual Twitter Analytics (Vista): Temporally changing sentiment and the discovery of emergent themes within sport event tweets. *Online Inf. Rev.* **2016**, *40*, 25–41. [CrossRef]
45. Lakhiwal, A.; Kar, A.K. Insights from twitter analytics: Modeling social media personality dimensions and impact of breakthrough events. In Proceedings of the 15th Conference on e-Business, e-Services and e-Society (I3E), Swansea, UK, 13–15 September 2016; Volume 9844, pp. 533–544.
46. De Maio, C.; Fenza, G.; Loia, V.; Parente, M.; Cuzzocrea, A. Towards OLAP Analysis of Multidimensional Tweet Streams. In Proceedings of the DOLAP '15: ACM Eighteenth International Workshop on Data Warehousing and OLAP, Melbourne, Australia, 19–23 October 2015; pp. 69–73.
47. Blachnik, M.; Kordos, M. Comparison of Instance Selection and Construction Methods with Various Classifiers. *Appl. Sci.* **2020**, *10*, 3933. [CrossRef]
48. Kowsari, K.; Meimandi, K.J.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text Classification Algorithms: A Survey. *Information* **2019**, *10*, 150. [CrossRef]
49. Salas-Rueda, R.-A. Percepciones de los estudiantes sobre el uso de Facebook y Twitter en el contexto educativo por medio de la ciencia de datos y el aprendizaje automático. *Pixel-Bit. Rev. Medios Educ.* **2020**, *58*, 91–115. [CrossRef]
50. Simeone, O. A Very Brief Introduction to Machine Learning with Applications to Communication Systems. *IEEE Trans. Cogn. Commun. Netw.* **2018**, *4*, 648–664. [CrossRef]
51. Wang, Z.; Bai, G.; Chowdhury, S.; Xu, Q.; Seow, Z.L. TwiInsight: Discovering Topics and Sentiments from Social Media Datasets. *Comput. Res. Repos.* **2017**, *2017*, 1–14.
52. Saura, J.R.; Reyes-Menendez, A.; Alvarez-Alonso, C. Do online comments affect environmental management? Identifying factors related to environmental management and sustainability of hotels. *Sustainability* **2018**, *10*, 3016. [CrossRef]

53. Valerio Ureña, G.; Serna Valdivia, R. Redes sociales y bienestar psicológico del estudiante universitario. *Rev. Electrónica Investig. Educ.* **2018**, *20*, 19. [CrossRef]
54. Alonso, M. Las redes sociales como canal de comunicación de las marcas de moda españolas. El caso de Zara, Mango y el Corte Inglés. *Index Comun.* **2015**, *5*, 77–105.
55. Razali, N.M.; Wah, Y.B. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *J. Stat. Model. Anal.* **2011**, *2*, 21–33.
56. Gujarati, D.N.; Porter, D.C. *Essentials of Econometrics*, 4th ed.; McGraw-Hill: New York, NY, USA, 2010.
57. Gureeva, A.N. Social Networks as a Media Communication Resource for Managing the Image of a Russian Higher Education Institution. *Mediascope* **2018**, *2*. [CrossRef]
58. Guzmán Duque, A.P.; del Moral Pérez, M.E.; González Ladrón de Guevara, F.; Gil Gómez, H. Impacto de twitter en la comunicación y promoción institucional de las universidades Impact of twitter on communication and institutional promotion of universities. *Pixel Bit. Rev. Medios Educ.* **2013**, *43*, 139–153. [CrossRef]
59. Hanifawati, T.; Ritonga, U.S.; Puspitasari, E.E. Managing brands' popularity on Facebook: Post time, content, and brand communication strategies. *J. Indones. Econ. Bus.* **2019**, *34*, 185. [CrossRef]
60. Wang, Y.; Youn, H. Feature weighting based on inter-category and intra-category strength for Twitter sentiment analysis. *Appl. Sci.* **2019**, *9*, 92. [CrossRef]
61. Andéhn, M.; Kazeminia, A.; Lucarelli, A.; Sevin, E. User-generated place brand equity on Twitter: The dynamics of brand associations in social media. *Place Brand. Public Dipl.* **2014**, *10*, 132–144. [CrossRef]
62. Morales, A.J.; Borondo, J.; Losada, J.C.; Benito, R.M. Efficiency of human activity on information spreading on Twitter. *Soc. Netw.* **2014**, *39*, 1–11. [CrossRef]
63. Saif, H.; He, Y.; Fernandez, M.; Alani, H. Contextual semantics for sentiment analysis of Twitter. *Inf. Process. Manag.* **2016**, *52*, 5–19. [CrossRef]
64. Kraiem, M.B.; Feki, J.; Khrouf, K.; Ravat, F.; Teste, O. Modeling and OLAPing social media: The case of Twitter. *Soc. Netw. Anal. Min.* **2015**, *5*, 1–15. [CrossRef]
65. Kraiem, M.B.; Feki, J.; Khrouf, K.; Ravat, F.; Teste, O. OLAP4Tweets: Multidimensional Modeling of tweets. In Proceedings of the 19th East-European Conference on Advances in Databases and Information Systems (ADBIS 2015), Poitiers, France, 8–11 September 2015.

applied sciences

MDPI

Article

Research on Sentiment Classification of Online Travel Review Text

Wen Chen [1,2], Zhiyun Xu [1], Xiaoyao Zheng [3], Qingying Yu [3] and Yonglong Luo [1,3,*]

[1] School of Geography and Tourism, Anhui Normal University, Wuhu 241003, China; wchen@ahnu.edu.cn (W.C.); zhyxu@mail.ahnu.edu.cn (Z.X.)
[2] School of Mathematics and Computer Science, Tongling College, Tongling 244000, China
[3] Anhui Provincial Key Laboratory of Network and Information Security, Anhui Normal University, Wuhu 241003, China; zxiaoyao@ahnu.edu.cn (X.Z.); ahnuyuq@ahnu.edu.cn (Q.Y.)
* Correspondence: ylluo@ustc.edu.cn; Tel.: +86-0553-5910645

Received: 19 June 2020; Accepted: 27 July 2020; Published: 30 July 2020

Abstract: In recent years, the number of review texts on online travel review sites has increased dramatically, which has provided a novel source of data for travel research. Sentiment analysis is a process that can extract tourists' sentiments regarding travel destinations from online travel review texts. The results of sentiment analysis form an important basis for tourism decision making. Thus far, there has been minimal concern as to how sentiment analysis methods can be effectively applied to improve the effect of sentiment analysis. However, online travel review texts are largely short texts characterized by uneven sentiment distribution, which makes it difficult to obtain accurate sentiment analysis results. Accordingly, in order to improve the sentiment classification accuracy of online travel review texts, this study transformed sentiment analysis into a multi-classification problem based on machine learning methods, and further designed a keyword semantic expansion method based on a knowledge graph. Our proposed method extracts keywords from online travel review texts and obtains the concept list of keywords through Microsoft Knowledge Graph. This list is then added to the review text to facilitate the construction of semantically expanded classification data. Our proposed method increases the number of classification features used for short text by employing the huge corpus of information associated with the knowledge graph. In addition, this article introduces online travel review text preprocessing, keyword extraction, text representation, sampling, establishment classification labeling, and the selection and application of machine learning-based sentiment classification methods in order to build an effective sentiment classification model for online travel review text. Experiments were implemented and evaluated based on the English review texts of four famous attractions in four countries on the TripAdvisor website. Our experimental results demonstrate that the method proposed in this paper can be used to effectively improve the accuracy of the sentiment classification of online travel review texts. Our research attempts to emphasize and improve the methodological relevance and applicability of sentiment analysis for future travel research.

Keywords: user generated content; sentiment analysis; classification; keyword extraction; text representation; sampling; machine learning; TripAdvisor

1. Introduction

Tourism research has entered the era of big data. Based on big data analysis, academia and industry are now better positioned to understand and explore tourist behavior and the tourism market. Li et al. [1] contend that big data analysis can provide sufficient data without introducing sampling bias, and can also make up for the sample size limitations encountered by the survey data, thereby enabling a better understanding of tourist behavior. Sivarajah et al. [2] argued that big data analysis can lead to new knowledge; subsequently, such analysis has become the mainstream method used to obtain useful information.

From blogs and social media posts to online travel review sites, user-generated content (UGC) is one of the most important data sources for big data. UGC comprises insightful feedback that is spontaneously provided by users. This feedback information is widely available at little to no cost and can also be easily obtained [3]. Such feedback also has potential commercial value in fields such as targeted advertising, customer–company relationships, and brand communication [4,5].

Online travel review sites, such as TripAdvisor, generate large amounts of text-based online travel review data, which constitute an important type of UGC [6]. Online review text data can help researchers and practitioners to correctly understand tourists' travel preferences and needs [7,8]. The opinions expressed in user-generated comments also play an important role in influencing the choices of potential tourists [9,10].

The characteristics of big data have complicated the process of knowledge extraction. The question of how to transform data into valuable knowledge has become crucial for big data applications [11,12]. Previous research into online reviews has mainly focused on the quantitative ratings provided on the website, ignoring the text of online reviews [3]. Ratings cannot provide any information about the specific product characteristics that visitors like or dislike, and such information is typically included in the review text [5,13]. In addition, many users are overwhelmed by the enormous amount of review information provided on travel online review sites. Researchers in other fields have also raised similar questions. Ali et al. [14] noted that while urban traffic congestion is rapidly increasing, a city's rating score is insufficient to provide accurate information; however, comments or tweets may help travelers and traffic managers to understand all aspects of the city. As a result, it is necessary to establish an effective mechanism to help users identify the main content and emotions embedded in the review text [15].

Human emotions and emotional reasoning are understood to be important factors that influence consumer decision-making [16]. This makes sentiment analysis an effective method for mining the connotations of online travel review texts. Text sentiment analysis methods can be divided into dictionary-based methods [17], machine learning methods [13,18], deep learning methods [19,20], and hybrids of the above methods [21,22]. Alaei et al. [8] contend that dictionary-based systems rely on the use of sentiment dictionaries and rule sets. Their article proposes that such methods are unable to adapt to the rapid increase in data volume in the era of big data, so it is necessary to develop more effective automated methods for sentiment analysis. Deep learning methods usually require a large amount of training data to fully realize their potential; this training data usually requires expensive class labeling [23]. Among machine learning methods, support vector machines (SVM) and naive Bayes are the most widely used in the tourism-related sentiment analysis context [13]. Compared with neural networks, SVM and naive Bayes require fewer class annotations to train the model [8]. Most studies on the subject have shown [18] that SVM-based sentiment analysis of text produces superior results relative to other machine learning methods. Kirilenko et al. [13] compared automatic text sentiment analysis classifiers with humans and evaluated whether various types of automatic classifiers are suitable for typical applications in the tourism, hotel, and marketing research contexts. The article argues that on difficult and noisy datasets, automatic classifiers achieve worse performance than humans. It can therefore be concluded that the existing sentiment analysis technology needs to be improved to enable the analysis of specific data.

Contemporary researchers have proposed many effective solutions to improve the performance of SVM in sentiment analysis. Successful feature extraction is one of the main challenges faced by machine learning methods [24]. Feature extraction can reduce information loss and achieve improved discrimination ability in sentiment classification [25] tasks. In their study of feature selection methods, Manek et al. [25] proposed a Gini index feature selection method based on SVM to carry out sentiment classification for a large movie review dataset. Ali et al. [26] proposed a robust classification technology based on SVM and fuzzy domain ontology (FDO), used for the recognition of comment features and the mining of semantic knowledge. Their experimental results showed that the integration of FDO and SVM greatly improves the accuracy of extracting comments and opinion words, as well as the accuracy of opinion mining. Parlar et al. [27] proposed a new feature selection method based on the query expansion term weighting method in the information retrieval context. This study uses four classifiers to compare their method with other widely used feature selection methods, thereby verifying their method's effectiveness. Zainuddin et al. [28] proposed a latent semantic analysis (LSA) and random projection (RP) feature selection method for the sentiment analysis of Twitter data, and thereby constructed a new Twitter mixed sentiment classification method. Kumar et al. [29] introduced swarm intelligence algorithms into the field of feature optimization in order to improve the sentiment classification performance accuracy. Pu et al. [30] used a variety of features to identify candidate opinion sentences, then used structured SVM to encode these opinion sentences for document sentiment classification. This article resolves the issue of sentiment classification problems arising when the sentiment of most sentences is inconsistent with the sentiment of the document overall.

As an effective feature selection method, semantic expansion has also been widely studied. Adhi et al. [31] designed a sentiment analysis model based on a naive Bayes classifier and the semantic extension method, proving that the semantic extension method can improve the accuracy of sentiment analysis. Fang et al. [32] integrated the context features extracted from the comment sentences and the external knowledge retrieved from the sentiment knowledge graph into a neural network to compensate for the lack of available training data, consequently obtaining better sentiment analysis results. At the same time, as an effective channel for semantic expansion, knowledge bases such as WordNet and ConceptNet are widely used in sentiment analysis in multiple languages. Alowaidi et al. [33] proposed using Arabic WordNet as an external knowledge base to enrich the representation of tweets due to the weakness of the bag of words model; the use of naive Bayes and SVM on the Arab Twitter dataset verified that this external knowledge base can be used to improve sentiment analysis accuracy. Asgarian et al. [34] used Persian WordNet to generate a review corpus, proving that sentiment dictionary quality plays a key role in improving the quality of sentiment classification in the Persian language. Moreover, Agarwal et al. [35] proposed a novel sentiment analysis model based on ConceptNet and common sense extracted from context information.

At the same time, a number of scholars in tourism research have studied the application of sentiment analysis to tourism and hospitality-related data. Several existing works [8,13] have already summarized the sentiment analysis methods adopted by the academic community in the tourism context prior to 2016; therefore, this article only summarizes the relevant literature published after 2017 in Table 1. Among these works, Höpken et al. [36] extracted customer feedback from two online platforms and carried out sentiment analysis and opinion mining, verifying that SVM is best able to solve the problem of sentiment analysis compared with other related methods. Akhtar et al. [37] used topic modeling technology to identify hidden information and other aspects, then performed sentiment analysis on classified hotel review text sentences. Ma et al. [38] performed sentiment analysis on TripAdvisor's review data using Leximancer. Ko et al. [39] applied statistical analysis methods to a large number of consumer review texts obtained from Expedia, enabling these authors to understand the experiences of hotel guests and analyze their association with satisfaction. Stepchenkova et al. [40] selected and compared three of the best-performing sentiment analysis methods to quantify respondents' views on travel in China. Bansal et al. [41] further proposed a sentiment classification method based on mixed attributes. By capturing implicit word relationships and combining domain-specific knowledge, these authors were able to obtain a fine-grained emotional orientation of online consumer reviews. Finally, Lawani et al. [42] used the AFINN dictionary (a lexicon based on unigrams) to extract the sentiments from comments left by Airbnb guests and derive a quality score from those comments.

An analysis of the above literature reveals that the academic community has carried out fruitful work in the field of sentiment analysis, particularly as regards the feature selection of SVM. Although these related topics have been extensively researched, certain specific types of content, such as online travel review texts for TripAdvisor, still present some challenges when using sentiment analysis [43]. This is because the key features of reviews vary significantly from site to site, meaning that it cannot be assumed that the sentiment analysis method and findings of a certain site will be applicable to all other review sites [44]. On the subject of the sentiment analysis of online travel review texts, most existing sentiment analysis models fail to comprehensively and effectively consider the data characteristics of travel review texts during the modeling process. Online travel review texts have their own inherent characteristics. Most review texts are short, which makes it difficult to extract keywords; in addition, the sentiment distribution of short texts is uneven [45] (for example, the texts with the highest and lowest scores are comparatively few). These characteristics make it difficult for accurate sentiment analysis results to be obtained for online travel review texts [46]. In addition, the accuracy of existing automated sentiment analysis methods is also low [13].

Table 1. The methods of sentiment analysis used in tourism research.

Reference	Methods	Data
[36]	Word list-based methods, supervised learning methods (including k nearest neighbors, support vector machines and naive Bayes)	TripAdvisor, Booking
[37]	Developed by the author	TripAdvisor
[38]	Leximancer	TripAdvisor
[39]	Statistical Analysis	Expedia
[40]	Deeply Moving, Pattern and SentiStrength	Survey data
[41]	Developed by the author	TripAdvisor, Amazon
[42]	AFINN	Airbnb

In order to deal with the sentiment analysis-related challenges brought about by the data features of online travel review texts, this study converted the sentiment analysis of online travel review texts into a multi-classification process based on machine learning methods, and further conducted research on sentiment classification methods for such texts. In order to improve the classification accuracy of online travel review texts, the current research mainly addresses the following problems related to previous research. The main contributions of the paper include:

- Based on the word similarity calculation results, the present study compares three keyword extraction methods and provides the most suitable keyword extraction method for online travel review text.
- After considering the sparse features of online travel review texts, this paper expands the semantics of text keywords based on Microsoft Knowledge Graph in order to build richer and more valuable classification features.
- To address the problem of uneven sentiment distribution in online travel review texts, two types of sampling methods are compared and the most suitable online travel review text sampling method is identified.
- This article introduces the online travel review text preprocessing method, Word2vec-based text representation method, classification label acquisition method, and machine learning method-based sentiment classification method, thereby presenting the entire sentiment analysis process of online travel review texts.
- TripAdvisor is a frequently used text source in sentiment analysis [47]; thus, by analyzing more than 20,000 review text datasets for four famous attractions in four countries derived from TripAdvisor, this paper validates the proposed method from a relatively extensive sample, which allows us to draw more reliable conclusions. Experimental results reveal that the method proposed in this paper is better suited to processing the sentiment classification of online travel review texts, and consequently provides a reference for related travel research.

2. Materials and Methods

Sentiment analysis generally includes multiple steps [48]. As can be seen from Figure 1, the sentiment analysis process proposed in this paper includes the following five steps:

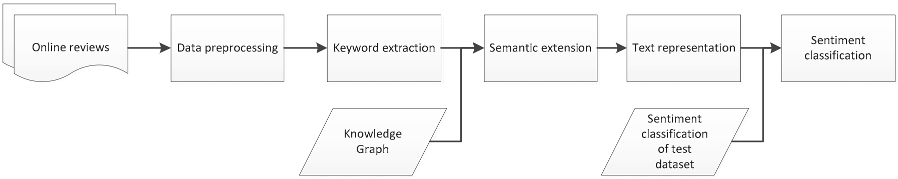


Figure 1. System framework.

(1) Data retrieval. In this study, a crawler program written in Python was used to obtain the texts, namely English descriptions of four famous attractions in four countries from the travel review website TripAdvisor, used as sentiment analysis data. This process is relatively simple; due to space limitations, it will not be described here.

(2) Data preprocessing. Section 2.1 introduces the steps involved in online comment text preprocessing.

(3) Keyword extraction and semantic expansion of comment texts. In order to improve classification accuracy, Section 2.2 introduces our online travel review text keyword extraction method and keyword semantic expansion method based on Microsoft Knowledge Graph.

(4) Text representation. Section 2.3 introduces the text representation method based on Word2vec.

(5) Sentiment classification. Section 2.4 introduces the sentiment classification method adopted in this paper.

2.1. Data Preprocessing

Not all characters included in the text of online travel reviews are important. For example, most reviews include words, punctuation, etc. that do not describe the subject of the text. Retaining all characters will lead to the formation of high-dimensional features; this will not only increase the time required for classification learning, but will also introduce a lot of noisy data into the classification and affect the classification accuracy. It is therefore necessary to preprocess the data. The preprocessing process used in this article comprises the following four steps:

1. Remove HTML tags
2. Remove non-letters
3. Convert words to lower case and split them
4. Remove stop words.

In step 1, Python's BeautifulSoup library was used to remove HTML tags such as '<br>' from the comment text. Steps 2–4 were implemented using NLTK (Natural Language Toolkit) [49] and regular expressions. Here, the second step deletes punctuation, numbers and other non-English characters from the comment text; the third step divides the sentence into words and converts all of these words to lower case; finally, the fourth step uses the stop word list provided by NLTK and deletes these words from the comment text and stoplist. The stop word list contains some noise words that do not describe the text subject ("the", "is", "are", "a", "an", etc.). In addition, combined with the characteristics of the dataset in this article, we added some specific vocabulary words (for example: "Mutianyu", "Great Wall", "China"). These specific high-frequency words will affect the subsequent keyword extraction and sentiment analysis results. However, these words are usually objective descriptions of scenic spots and accordingly do not help with the sentiment analysis.

2.2. Keyword Extraction and Semantic Expansion

The online travel review text obtained in this article pertains to multiple attractions. As shown in Figures 2–5, before preprocessing, the length of the review text about Mutianyu Great Wall, Beijing, China is mostly between 260 and 280 words. Moreover, the length of the comment text for the Harry Potter Wizarding World Theme Park in Orlando, USA is between 90 and 130 words; the comment text for the Tower of London, England is between 90 and 140 words in length; and the lengths of the comment text for the Sydney Opera House in Australia are mostly in two categories (90 to 120 and 200 to 300 words). Because preprocessing will delete some characters that are not related to sentiment classification, the text will be shorter after preprocessing. It is difficult to extract effective feature words from shorter text and thus more difficult to obtain better sentiment classification results [46]. In order to improve the effectiveness of sentiment classification for online travel review text, this paper proposes a keyword semantic expansion method based on knowledge graphs. First, we compared several keyword extraction methods and selected the TextRank method as having the best effect [50] for achieving keyword extraction for online travel review text. Secondly, through the use of Microsoft Knowledge Graph, a conceptual list of keywords for each comment was obtained. This concept list of keywords can be used to expand the semantics of the comment text and provide a richer and more valuable classification feature for the classifier. Next, the specific implementation steps will be introduced.

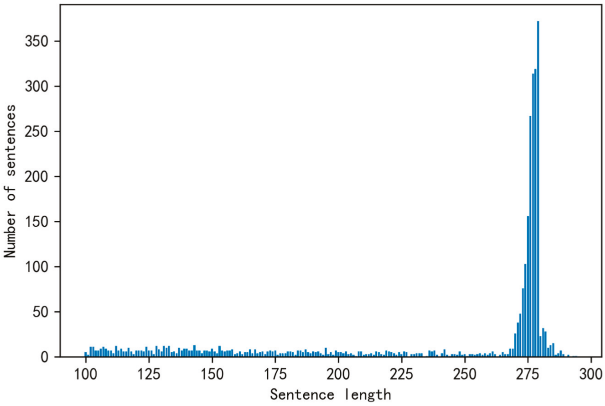


Figure 2. Length of the comment text of Mutianyu Great Wall in Beijing, China.

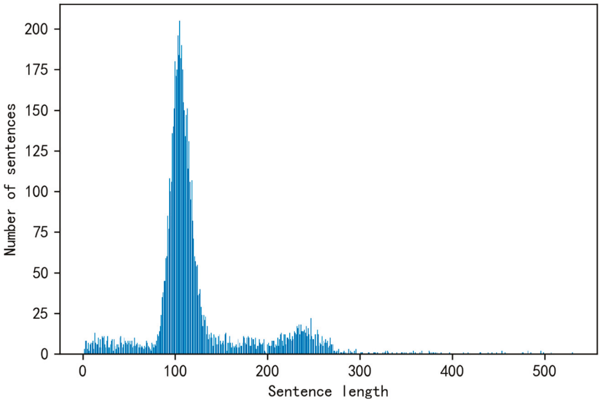


Figure 3. Length of the comment text in the Harry Potter Wizarding World Theme Park in Orlando, USA.

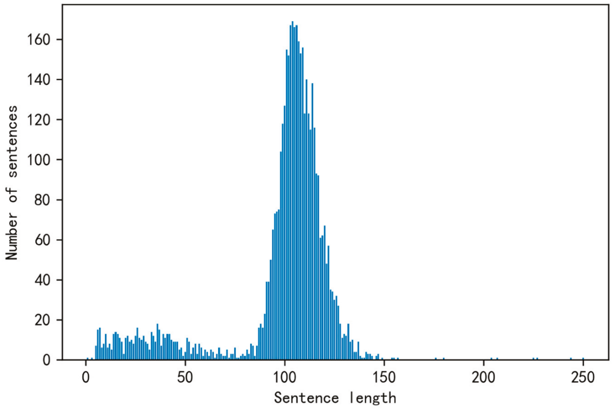


Figure 4. Length of the comment text of the Tower of London.

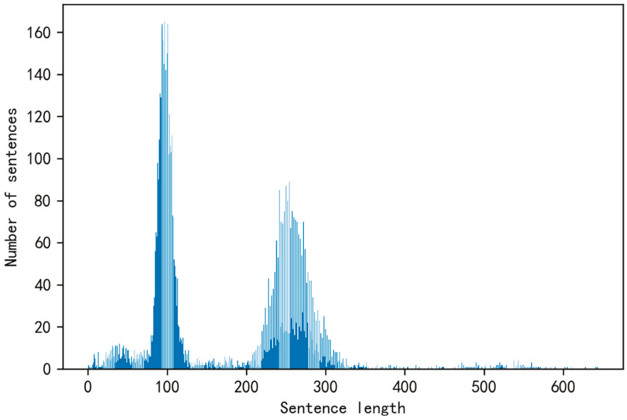


Figure 5. Length of the comment text of the Sydney Opera House in Australia.

(1) Keyword extraction

Text keyword extraction is a machine learning algorithm-based text feature extraction method. In fields such as text-based recommendation and search, the accuracy of text keyword extraction is directly related to the final effect. Accordingly, text keyword extraction is an important research direction in the field of text mining. Text keyword extraction methods can be divided into supervised, semi-supervised, and unsupervised methods [51]. Supervised and semi-supervised methods regard keyword extraction as a classification problem and require a labeled training corpus to train the keyword extraction model. However, for massive datasets, labeling the training corpus is often very time-consuming. For its part, the unsupervised keyword extraction method does not require a manually annotated corpus, and is therefore more suitable for the keyword extraction of massive comment texts [52].

The TextRank algorithm proposed by Mihalcea et al. [50] draws on the realization of PageRank, which is the core algorithm of Google search. This is an unsupervised keyword extraction method. Unlike TF-IDF (term frequency—inverse document frequency), LDA (Latent Dirichlet Allocation), etc., TextRank divides the text into several units (e.g., words, sentences) and builds a graph model; keyword extraction can thus be achieved using only the information contained in a single document.

The process by which TextRank extracts text keywords comprises the following steps:

(1) Divide the given text into sentences.

(2) For each sentence segmentation and part-of-speech tagging, filter out stop words, so that only words belonging to the specified part-of-speech are reserved as candidate keywords.

(3) Construct the candidate keyword graph G = (V, E), where V is the node set comprising the candidate keywords generated in step (2); next, use the co-occurrence relationship to construct the edges between any two points.

(4) Calculate the weight of each node. These node weights are sorted in reverse order so that the most important words are obtained as candidate keywords.

(5) Mark the candidate keywords obtained in step (4) in the original text; if adjacent phrases are formed, these are combined into multi-word keywords.

A variety of keyword extraction algorithms represented by TextRank are widely used in tourism and many other fields. Shouzhong et al. [53] integrates TF-IDF and TextRank to mine and analyze personal interests from Weibo text. Paramonov et al. [54] developed a new method combining well-known keyword extraction algorithms (e.g., TextRank and Topic PageRank) and a thesaurus-based procedure, thereby improving the connectivity of the text-via-keyphrase graph while also increasing the accuracy and recall rate of key phrase extraction. Gagliardi et al. [55] integrated the word embedding model and clustering algorithm to establish a novel method capable of automatically extracting keywords/phrases from text without supervision. Ali et al. [56] used the N-gram method to extract the risk factors of heart disease diagnosis and applied these to an intelligent heart disease prediction system, improving the accuracy of heart disease diagnosis.

In Section 3.2, based on the similarity calculation results of the words, and following experiments with TF-IDF and LDA, it is determined that the keywords extracted by TextRank are more suitable for ascertaining the actual semantics of online travel text reviews. Therefore, this study used TextRank for text keyword extraction purposes.

(2) Keyword semantic expansion

Text feature semantic expansion is an effective method of solving the sparse text problem [57]. Wang et al. [58] conceptualized short text into a set of concepts and embedded the original text in order to form word vectors. Experimental results verify that the convolutional neural network based on this word vector can achieve good short text classification results. Rosso et al. [59] believe that combining large-scale unstructured content (text) and high-quality structured data (knowledge graph) can improve text analysis.

Microsoft Knowledge Graph [60] has learned a large amount of common sense knowledge through learning from billions of web pages and years of search logs. The system-provided conceptual model maps text entities into semantic concept categories with specific probabilities; for example, “Microsoft” may automatically map to “software companies” and “Fortune 500 companies” [61]. This paper introduces the conceptual model of the Microsoft Knowledge Graph to expand the semantics of online travel review text keywords. This knowledge graph-based keyword semantic expansion method utilizes the huge information corpus of the Microsoft knowledge graph to expand the semantics of the text. This method overcomes the issue of fewer features being available that is caused by the sparseness of short texts, and accordingly provides richer and more valuable classification features for short text sentiment classification. We demonstrate the improvement in classification accuracy brought about by this method in the experiment discussed in Chapter 3.

2.3. Text Representation

(1) Text representation of comments based on Word2vec

Representing text as structured data that is able to be handled by machine learning classification algorithms is a highly important part of the text classification process. In 2013, Google released the software tool Word2vec for training word vectors [62]. Word2vec’s high-dimensional vector model solves the multi-dimensional semantic problem, because it can quickly and effectively express words in high-dimensional vector form through the optimized training model according to a given corpus, thereby providing a new tool for the application research in the field of natural language processing [63]. Academic research [64,65] demonstrates that Word2vec has achieved excellent performance in the fields of text similarity calculation and text classification. In light of the above analysis, this study opted to construct Word2vec vectors for the pre-processed and semantically expanded comment text.

(2) Data normalization

Normalized data exhibits enhance stability for attributes with very small variances, while maintaining 0 entries in the sparse matrix [62]. Therefore, this study used the normalization method to scale the text vector represented by Word2vec to between 1 and 0. The formula utilized is as follows:

$$x_i{}' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

In Equation (1), $x_i{}'$ represents the result of normalization, while x_i represents the data that needs to be normalized. Moreover, x_{max} and x_{min} represent the maximum and minimum values in the dataset, respectively.

2.4. Sentiment Classification

For massive texts, one effective solution involves transforming sentiment analysis into classification and applying machine learning methods in order to solve such problems [66]. This article has introduced the problems encountered by deep learning methods, along with the excellent results achieved by machine learning methods in the text sentiment analysis context. Therefore, using the online travel review text data processed in Sections 2.1–2.3 as the training data, this SVM was chosen in this study as the method of sentiment classification. In Section 3.4, through the analysis of experimental results, the most suitable sentiment classification model for processing online travel review texts is then provided.

3. Case Study

This section introduces the research process utilized in this article and draws conclusions from a sentiment classification experiment on online tourist review texts of multiple attractions. In more detail, Section 3.1 describes the experimental dataset and the results of preprocessing; Section 3.2 introduces the experimental process of keyword semantic expansion based on the knowledge graph; Section 3.3 introduces the text representation based on Word2vec; finally, Section 3.4 introduces the sentiment classification based on SVM experiments and result analysis.

3.1. Data Acquisition and Preprocessing Experiment

As shown in Table 2, the present research used a crawler program written in Python to obtain four datasets from the TripAdvisor website. Mutianyu_Great_Wall contains review text pertaining to Mutianyu Great Wall, which is the number one attraction on the TripAdvisor website in Beijing, China. It contains a total of 2772 pieces of review data in English published by tourists from January 2016 to December 2019. Wizarding_World_of_Harry_Potter contains comment text pertaining to the Harry Potter Wizarding World Theme Park in Orlando, USA, specifically, 6641 pieces of comment data published by tourists from June 2017 to December 2019 in English. Tower_of_London contains comment text about the Tower of London in the United Kingdom. It contains the data of 4428 comments in English published by tourists from July 2018 to December 2019. Finally, Sydney_Opera_House contains review text pertaining to the Sydney Opera House in Australia. It contains 6776 pieces of comment data in English published by tourists from March 2017 to December 2019.

Table 3 presents a piece of comment text in the Mutianyu_Great_Wall dataset and its pre-processed results. In the table, the first column is the comment text published by tourists, while the second column is the pre-processed text. As can be seen from the introduction in Section 2.2, the preprocessing operation removes any original comment text content that is unrelated to sentiment classification.

Table 2. Experimental data set.

Dataset	URL	Size
Mutianyu_Great_Wall	https://www.tripadvisor.cn/Attraction_Review-g294212-d325811-Reviews-Mutianyu_Great_Wall-Beijing.html	2772
Wizarding_World_of_Harry_Potter	https://www.tripadvisor.cn/Attraction_Review-g34515-d1968468-Reviews-The_Wizarding_World_of_Harry_Potter-Orlando_Florida.html	6641
Tower_of_London	https://www.tripadvisor.cn/Attraction_Review-g186338-d187547-Reviews-Tower_of_London-London_England.html	4428
Sydney_Opera_House	https://www.tripadvisor.cn/Attraction_Review-g255060-d257278-Reviews-Sydney_Opera_House-Sydney_New_South_Wales.html	6776

Table 3. Review and preprocessed results.

Review	Preprocessed Review
If you want to go individually, not with organized tour, we were offered the option to go by taxi. The taxi left us to the bus station we bought tickets for the bus, the entrance and the cable. The bus left us at the cable station and the cable took us to the Wall. It was very easy and not so expensive and we arranged for the hours which were convenient for us.	individually organized tour offered option taxi taxi left bus station bought tickets bus entrance cable bus left cable station cable easy expensive arranged hours convenient

3.2. Keyword Semantic Expansion Experiment Based on Knowledge Graph

(1) Comparison of text keyword extraction methods

In this study, three types of text keyword extraction methods, namely TF-IDF, LDA, and TextRank were selected to carry out comparative experiments. Taking the comment text in Table 3 as an example, the manually provided subject term is “transport”, while the second column of Table 4 presents the keywords with the largest calculation result values obtained by each of the three methods. Among them, the calculation results of the two keywords obtained by TF-IDF are the same.

Table 4. Results of three types of keyword extraction methods.

Method	Keyword	Similarity Result
TF-IDF	cable, bus	cable: 0.139 bus: 0.379
LDA	offered	offered: −0.022
TextRank	bus	bus: 0.379

Word2vec is able to convert words into vectors and calculate the distance between the vectors. The larger the value of the calculation result, the greater the similarity between the two words [63]. Based on Word2vec’s word vector similarity calculation, this study calculated the similarity of the word vector using the first-order keywords obtained by the three methods in addition to the subject words (“transport”) of the manually provided comment text. The calculation results are shown in the third column of Table 4. Here, the keywords obtained by TextRank are the most relevant to the subject words of the manually provided review text. TF-IDF also identified the most relevant keyword, ”bus”. However, the keyword “cable”, which has the same weight as “bus”, has poor relevance to the subject words of the manually provided review text, which affects the final result. LDA requires a large corpus (i.e., large amount of comment text) for accurate results to be obtained. However, this research requires keywords to be derived for each short text of the comment. Therefore, LDA is unsuitable for this research, and the final keyword extraction effect is also poor. This study randomly selected 10% of the samples in each dataset and used the above three methods to extract keywords.

Following experimental comparison, TextRank was found to have the best keyword extraction effect. Therefore, TextRank was used to extract the text keywords from online travel reviews.

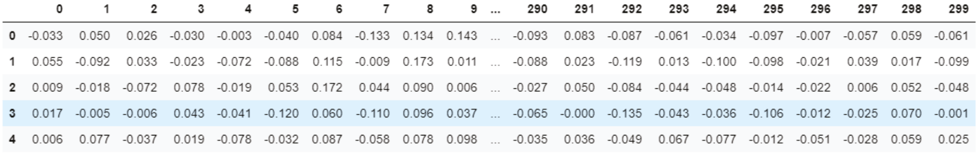
(2) Keyword semantic expansion experiment

This study obtained a concept list of online travel review text keywords using the conceptual model of Microsoft Knowledge Graph [67]. For example, the conceptual list of the keyword “bus” in the comment text of Table 3 is as follows: vehicle, public transportation, large vehicle, etc.

For the four datasets listed in Table 2, TextRank was used to extract text keywords from a total of 20,617 comment texts. In the next step, the concept list for the first-ranked keywords was obtained in ascending order of weight. Although Microsoft Knowledge Graph covers a very wide range, it does not cover any word. Following calculation, Microsoft Knowledge Graph returned results for 97.6% of the keywords in this experiment. Finally, we added the return results of each keyword to the pre-processed comment text in order to create a pre-classification dataset.

3.3. Text Representation and Normalization Experiment

Once preprocessing and semantic expansion was complete, the comment text was typically under 300 characters in length. Therefore, googlenews-vecctors-negative300.bin [62], a word vector library of news corpora pre-trained by Google, was selected to create a comment text vector. The final results are illustrated in Figure 6. Each line in the figure is a normalized 300-dimensional Word2vec real vector, which represents a specific comment text.



	0	1	2	3	4	5	6	7	8	9	...	290	291	292	293	294	295	296	297	298	299
0	-0.033	0.050	0.026	-0.030	-0.003	-0.040	0.084	-0.133	0.134	0.143	...	-0.093	0.083	-0.087	-0.061	-0.034	-0.097	-0.007	-0.057	0.059	-0.061
1	0.055	-0.092	0.033	-0.023	-0.072	-0.088	0.115	-0.009	0.173	0.011	...	-0.088	0.023	-0.119	0.013	-0.100	-0.098	-0.021	0.039	0.017	-0.099
2	0.009	-0.018	-0.072	0.078	-0.019	0.053	0.172	0.044	0.090	0.006	...	-0.027	0.050	-0.084	-0.044	-0.048	-0.014	-0.022	0.006	0.052	-0.048
3	0.017	-0.005	-0.006	0.043	-0.041	-0.120	0.060	-0.110	0.096	0.037	...	-0.065	-0.000	-0.135	-0.043	-0.036	-0.106	-0.012	-0.025	0.070	-0.001
4	0.006	0.077	-0.037	0.019	-0.078	-0.032	0.087	-0.058	0.078	0.098	...	-0.035	0.036	-0.049	0.067	-0.077	-0.012	-0.051	-0.028	0.059	0.025

Figure 6. Vector representation of text.

3.4. Sentiment Classification Experiment

(1) Acquisition of training set classification labels

The machine learning classification method represented by SVM requires training data with sentiment classification results for model training. The sentiment classification results for these training data are also referred to as the training set classification labels. In this study, manual analysis and sentiment analysis software were used to generate the classification labels for the training set.

SentiStrength [68] is a software package that estimates the strength of positive and negative emotions contained in text. It also has an artificial level of accuracy for short social network texts in English. We chose the nine-level sentiment classification results provided by SentiStrength. For negative emotions, the scores range from −1 (not negative) to −4 (extremely negative); for positive emotions, moreover, the scores range from 1 (not positive) to 4 (extremely positive); 0 represents neutral emotion. In this study, the SentiStrength results were again scored by humans, and the adjustment rate was about 24.7%. Finally, the sentiment analysis results of the dataset are presented in Figures 7–10. The abscissa represents the sentiment analysis results, which range from −4 to 4 in a total of nine categories; the ordinate indicates the number of samples in each category. It can be seen that the number of samples of each sentiment value is extremely uneven. For example, in the review dataset for the Mutianyu Great Wall in Beijing, China, the number of texts in category 2 is 1266, while there is only 1 text in category −4. Moreover, there is no category −4 data in the review data of the Sydney Opera House in Australia.

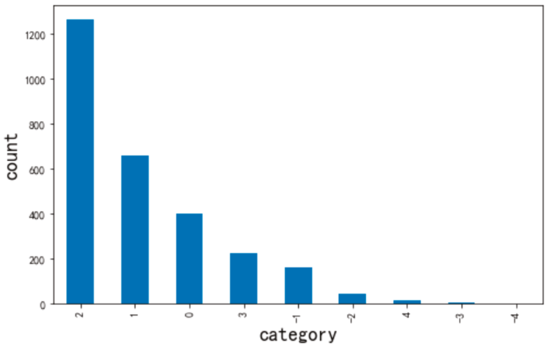


Figure 7. The sentiment distribution of the review text of Mutianyu Great Wall in Beijing, China.

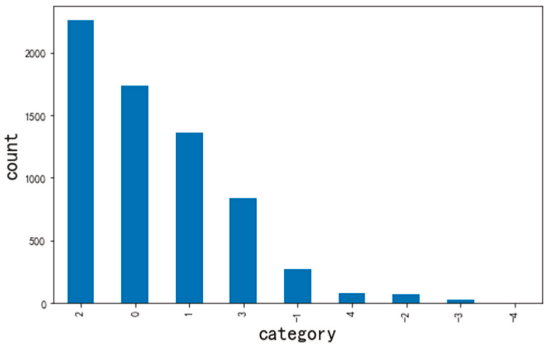


Figure 8. The sentiment distribution of the review text of the Harry Potter Wizarding World Theme Park in Orlando, USA.

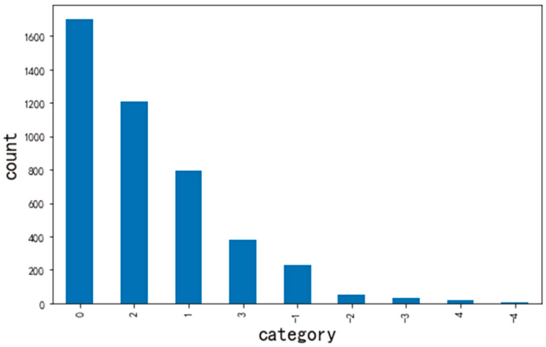


Figure 9. The sentiment distribution of the review text of Tower of London.

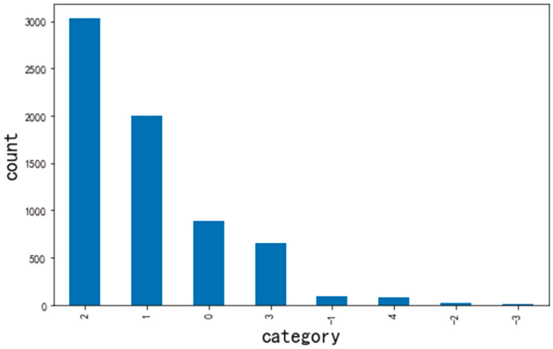


Figure 10. The sentiment distribution of the review text of Sydney Opera House, Australia.

(2) Sampling experiment of unbalanced data

From the analysis presented in the previous section, we can see that the sentiment distribution of online travel review texts is very uneven. In fact, this is a typical unbalanced dataset. For unbalanced datasets, machine learning classifiers will tend to incorrectly divide new samples into categories with more samples, resulting in classification errors [69]. The methods used to process unbalanced datasets are mainly divided into undersampling, oversampling, and improved methods [70]. This study used Python to implement two types of sampling methods. Our experimental results demonstrate that, due to the extremely uneven sentiment distribution of the experimental dataset used, the undersampling dataset was so small that it was difficult to obtain more accurate classification results. Overall, Naive Random Over Sampler (ROS) [71] achieved the best sampling results.

(3) Evaluation index

The evaluation indicators of classification results that have been adopted by academia include Accuracy, Precision, Recall, and F1 score [72]. In binary classification, the sample categories are divided into positive and negative types. Let us suppose that TP represents the number of samples that are both actually positive and classified as positive, while FP denotes the number of samples that are actually negative but classified as positive; moreover, FN represents the number of samples that are in fact positive but are classified as negative, while TN indicates the number of samples that are both actually negative and classified as negative. In addition, the accuracy rate refers to the proportion of correct samples classified as positive to the samples classified as positive. The calculation formula for this is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Furthermore, the recall rate refers to the proportion of correct samples classified as positive to actually positive samples, and the calculation formula is as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Finally, the F1 score is the harmonic average of the precision rate and the recall rate. The calculation formula is as follows:

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

The accuracy rate reflects the model's ability to distinguish negative samples: the higher the accuracy rate, the stronger the model's ability to distinguish negative samples. Moreover, the recall rate reflects the model's ability to identify positive samples: the higher the recall rate, the stronger the model's ability to recognize positive samples. In addition, the F1 score is the combination of the accuracy rate and recall rate: the higher the F1 score, the more robust the model. While accuracy is the simplest and most intuitive evaluation index in classification, it is also affected by obvious defects. For example, if we assume that 99% of the samples are positive samples, the classifier could obtain 99% accuracy if it always predicted a positive result, but its actual performance would be very low. That is to say, when the proportion of samples in different categories is highly uneven, the category with the largest proportion often becomes the most important factor affecting the accuracy. As the experimental data in this study was unbalanced data, we did not use the accuracy rate as a classification result evaluation index. Instead, we selected three indicators—accuracy rate, recall rate, and F1 score—to measure the classification results.

(4) SVM-based sentiment classification

Python's sklearn was used to implement the SVM algorithm. After a large number of experiments, the kernel function RBF (Radial Basis Function) was found to achieve the highest classification accuracy, while other parameters were assigned default values. We used 30% of the data as test data and the remaining 70% as training data.

The comparative experimental results of one dataset (Mutianyu_Great_Wall) are presented in Table 5. The first row of Table 5 displays the classification results of SVM. The classification accuracy of SVM on this imbalanced dataset is very low, as it assigns most of the samples to the category with the largest number of samples. Once ROS sampling and the Word2vec vectorization of text is complete, the data in the second row of Table 5 shows that the SVM algorithm's classification result has been greatly improved. The next experiment carried out involves extracting TextRank keywords from the comment text and expand the semantics of the keywords with the largest weights based on the Microsoft Knowledge Graph. The semantic expansion of keywords and pre-processed online travel review text make up the SVM classification dataset. Moreover, the experimental results in the third row of Table 5 list the final classification results; it can be seen from this table that the knowledge graph-based keyword semantic expansion method proposed in this paper optimizes the classification results.

Table 5. Mutianyu_Great_Wall experiment results.

Method	Precision	Recall	F1 score
SVM	0.108	0.238	0.148
SVM + ROS + Word2vec	0.840	0.822	0.830
SVM + ROS + Word2vec + KnowledgeGraph	**0.859**	**0.901**	**0.879**

Optimal solution in the comparison result is marked in bold.

Table 6 presents the experimental results of the other three datasets. Similar to the experimental results of the Mutianyu_Great_Wall dataset, it can be seen that the sampling technique, Word2vec-based text vectorization, and knowledge graph-based keyword semantic expansion method effectively improve the classification effect. Similar experimental results obtained on different datasets verify the universality of this method. In short, this provides an effective solution for sentiment analysis of online travel review texts.

Table 6. Experimental results of the other three data sets.

Method	Wizarding_World_of_ Harry_Potter			Tower_of_London			Sydney_Opera_House		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
SVM	0.108	0.339	0.164	0.701	0.404	0.512	0.287	0.445	0.348
SVM + ROS + Word2vec	0.637	0.643	0.639	0.833	0.868	0.85	0.702	0.716	0.708
SVM + ROS + Word2vec + KnowledgeGraph	**0.823**	**0.843**	**0.832**	**0.933**	**0.944**	**0.938**	**0.820**	**0.837**	**0.828**

Optimal solution in the comparison result is marked in bold.

The receiver operating characteristic curve (ROC) is an evaluation method that demonstrates the accuracy of classification through intuitive graphics. Figures 11–14 show the ROC curves of the four data sets. We have labeled each sentiment category (the Sydney_Opera_House dataset has only eight sentiment categories) with a different color. The abscissa in the figures indicates the proportion of samples classified as positive but actually negative to all negative samples; the ordinate represents the proportion of all positive samples that are predicted to be positive and actually positive. The closer the ROC curve is to the upper left corner, the higher the accuracy of the experiment.

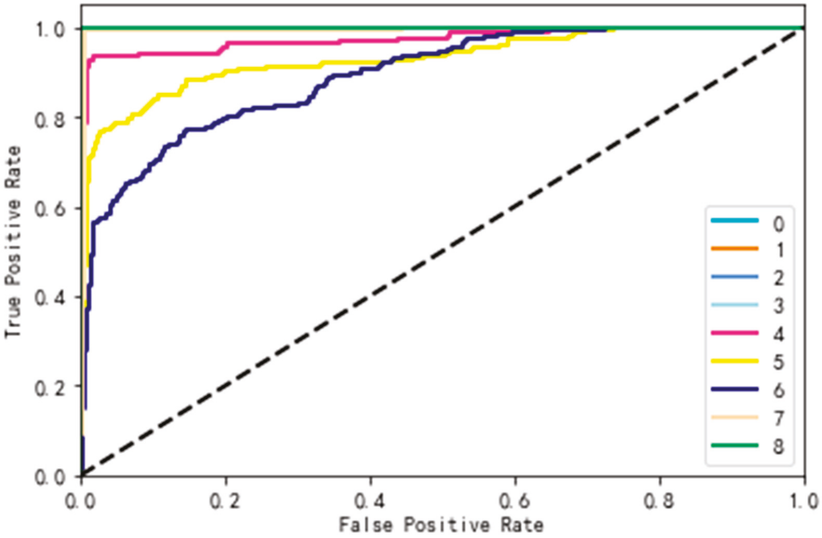


Figure 11. Mutianyu_Great_Wall receiver operating characteristic curve.

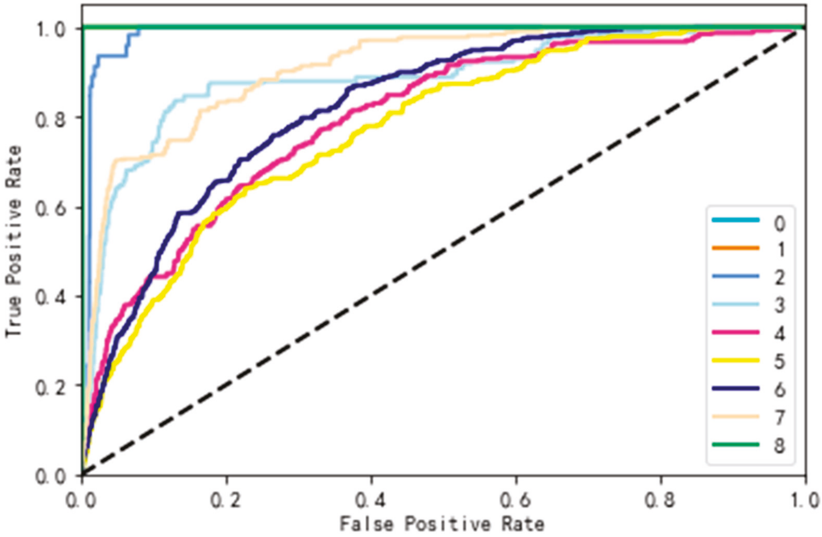


Figure 12. Wizarding_World_of_Harry_Potter receiver operating characteristic curve.

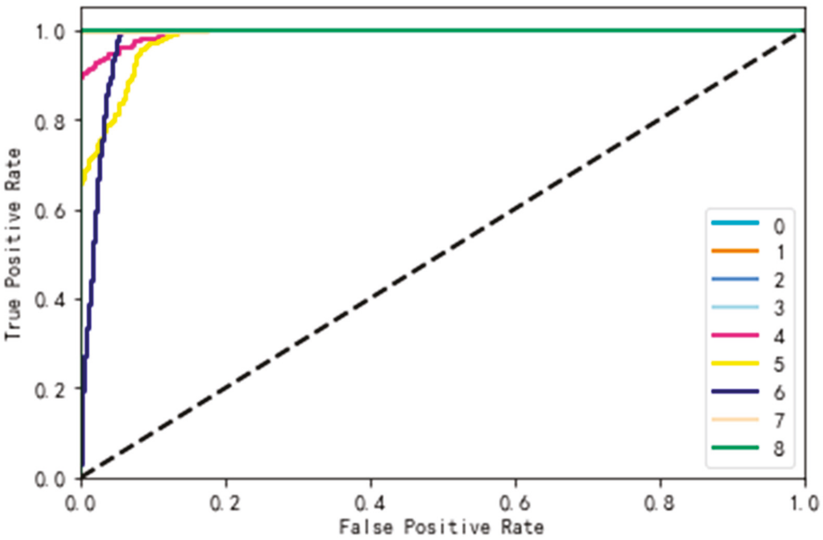


Figure 13. Tower_of_London receiver operating characteristic curve.

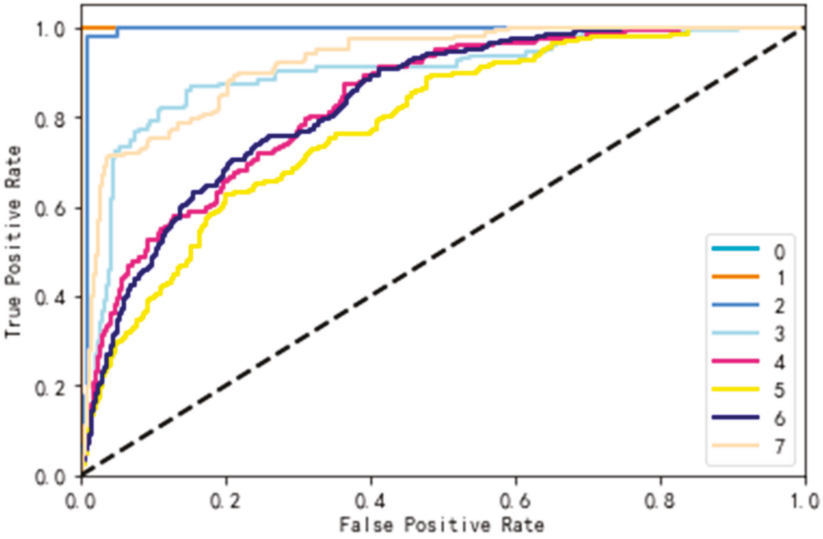


Figure 14. Sydney_Opera_House receiver operating characteristic curve.

4. Discussion

Sentiment analysis is a mainstream technology that employs social media analysis strategies to analyze customer feedback and comments. Conducting sentiment analysis based on websites such as TripAdvisor is desirable because a large number of free datasets can be obtained from such websites for large-scale research, while such large-scale data cannot easily be obtained via traditional research methods. Big data provides a new type of data for use in tourism research, and also puts forward higher requirements for data processing. Currently, few studies have been conducted on the applicability and accuracy of sentiment analysis methods in the tourism research literature. In addition, contemporary research ignores the possibility of integrating human knowledge, such as knowledge graphs, into existing methods in order to improve the text sentiment analysis performance. Big data is characterized by a huge data volume, and the speed and accuracy requirements for sentiment analysis are becoming steadily higher [8]. Therefore, the prospect of developing suitable and efficient sentiment analysis methods for specific types of big data in the tourism context is a highly valuable proposition.

The obtained sentiment analysis results based on TripAdvisor review text can be applied to multiple fields. For example, they can help sightseeing spots, restaurants, or hotels to explain comments and adopt corresponding countermeasures, which can in turn provide decision makers and customers with better decision-making information. Similarly, this approach can also be used to study theoretical issues related to customer satisfaction (for example, whether a tour guide service would improve the tourist experience). However, existing studies [43,44] have found that the key features of the review text differ substantially depending on which websites they are drawn from, and that it is therefore necessary to conduct sentiment analysis research on one specific website at a time. Therefore, research into machine learning sentiment analysis methods for TripAdvisor review texts will aid in the development of tourism research utilizing these texts. Compared with vocabulary sentiment analysis, one of the advantages of machine learning sentiment analysis is that it does not require humans to create a dictionary; this is beneficial because the production of such a dictionary is a time-consuming and laborious process. In addition, machine learning methods achieve more accurate performance on larger amounts of training data than can be obtained using vocabulary sentiment analysis [8]. Feature extraction is a key issue in the application of machine learning to the field of sentiment analysis [24]. Accordingly, this study designed and implemented a sentiment classification method based on the semantic expansion of text keywords that both increases the classification features and improves the accuracy of sentiment analysis, thereby providing a novel solution for machine learning sentiment analysis.

In terms of the specific details of the work of this article, in order to improve the accuracy of sentiment analysis conducted on online travel review texts, this study conducted extensive research work on the classification problems caused by the data features of online review texts. First, most online review texts are short texts, which makes it difficult to obtain more accurate sentiment classification results. To solve this problem, we designed a text keyword semantic expansion method based on a knowledge graph. In this part of the research, the present study compared three typical text keyword extraction methods and provided keyword extraction methods that are suitable for online travel review texts. In addition, based on Microsoft Knowledge Graph, the semantics of text keywords were expanded, and richer and more valuable sentiment classification features were constructed. The second part of the research involved comparing the two types of sampling methods and identifying which of these is more suitable for use in solving the uneven sentiment distribution problem in online review texts. This article fully describes the key aspects of online travel review text sentiment classification, establishes an effective sentiment classification research framework for online travel review text, and validates the proposed method based on a relatively extensive sample.

The work put forward in this paper aims to emphasize and improve the methodological relevance and applicability of sentiment analysis. However, there are some limitations:

- Studies have shown [10] that deleting comment text without emotional content can improve the accuracy of emotional classification. This idea is worth examining in future.
- In terms of keyword selection, this study only selected the keyword with the largest TextRank value. The questions of how to choose keywords with the same value, whether more keywords can be introduced for semantic expansion, the relationship between these choices, and the accuracy of sentiment classification are also worthy of further study.
- In the Word2vec-based text representation method, the use of different Word2vec corpora will yield different results. The best approach would be to train a corpus of specific topics [73].
- In terms of automated classification methods, studies have shown that the combination of LSTM (Long Short-Term Memory) and attention mechanisms [74] has resulted in excellent emotion

classification results. However, the question of whether these novel methods are suitable for the research object of this article is worthy of further study.

- In terms of experimental subjects, this article only studies English reviews from TripAdvisor, and does not investigate other online travel platforms and other languages. Therefore, it is highly advisable to investigate data in other languages and other platforms to verify the applicability of this method.

Author Contributions: Conceptualization, W.C., Z.X. and Y.L.; Methodology, W.C.; Software, W.C.; Validation, W.C., Q.Y. and X.Z.; Data curation, W.C.; Writing—original draft preparation, W.C.; Writing—review and editing, W.C., Z.X. and Y.L.; Supervision, Y.L.; Project administration, Y.L.; Funding acquisition X.Z., Q.Y. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grants 61972439, 61672039, 61702010, 61772034.

Acknowledgments: The authors would like to acknowledge all of reviewers and editors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, X.; Pan, B.; Law, R.; Huang, X. Forecasting tourism demand with composite search index. *Tour. Manag.* **2017**, *59*, 57–66. [CrossRef]
2. Sivarajah, U.; Kamal, M.M.; Irani, Z.; Weerakkody, V. Critical analysis of Big Data challenges and analytical methods. *J. Bus. Res* **2017**, *70*, 263–286. [CrossRef]
3. Guo, Y.; Barnes, S.J.; Jia, Q. Mining meaning from online ratings and reviews: Tourist satisfaction analysis using latent dirichlet allocation. *Tour. Manag.* **2017**, *59*, 467–483. [CrossRef]
4. Chuang, S. Co-creating social media agility to build strong customer-firm relationships. *Ind. Mark. Manag.* **2020**, *84*, 202–211. [CrossRef]
5. Kauffmann, E.; Peral, J.; Gil, D.; Ferrandez, A.; Sellers, R.; Mora, H. A framework for big data analytics in commercial social networks: A case study on sentiment analysis and fake review detection for marketing decision-making. *Ind. Mark. Manag.* **2019**, in press. [CrossRef]
6. Li, J.; Xu, L.; Tang, L.; Wang, S.; Li, L. Big data in tourism research: A literature review. *Tour. Manag.* **2018**, *68*, 301–323. [CrossRef]
7. Fang, B.; Ye, Q.; Kucukusta, D.; Law, R. Analysis of the perceived value of online tourism reviews: Influence of readability and reviewer characteristics. *Tour. Manag.* **2016**, *52*, 498–506. [CrossRef]
8. Alaei, A.; Becken, S.; Stantic, B. Sentiment Analysis in Tourism: Capitalizing on Big Data. *J. Travel Res.* **2019**, *58*, 175–191. [CrossRef]
9. Asghar, M.Z.; Kundi, F.M.; Ahmad, S.; Khan, A.; Khan, F. T-SAF: Twitter sentiment analysis framework using a hybrid classification scheme. *Expert Syst.* **2018**, *35*, e12233. [CrossRef]
10. Afzaal, M.; Usman, M.; Fong, A.; Fong, S. Multiaspect-based opinion classification model for tourist reviews. *Expert Syst.* **2019**, *36*, e12371. [CrossRef]
11. Gunther, W.; Mehrizi, M.H.R.; Huysman, M.; Feldberg, F. Debating big data: A literature review on realizing value from big data. *J. Strateg. Inf. Syst.* **2017**, *26*, 191–209. [CrossRef]
12. Mariani, M.M.; Baggio, R.; Fuchs, M.; Hoepken, W. Business intelligence and big data in hospitality and tourism: A systematic literature review. *Int. J. Contemp. Hosp. Manag.* **2018**, *30*, 3514–3554. [CrossRef]
13. Kirilenko, A.P.; Stepchenkova, S.; Kim, H.; Li, X. Automated Sentiment Analysis in Tourism: Comparison of Approaches. *J. Travel Res.* **2018**, *57*, 1012–1025. [CrossRef]
14. Ali, F.; Kwak, D.; Khan, P.; Islam, S.M.R.; Kim, K.; Kwak, K.S. Fuzzy ontology-based sentiment analysis of transportation and city feature reviews for safe traveling. *Transp. Res. Part C Emerg.* **2017**, *77*, 33–48. [CrossRef]
15. Luo, Y.; Tang, R. Understanding hidden dimensions in textual reviews on Airbnb: An application of modified latent aspect rating analysis (LARA). *Int. J. Hosp. Manag.* **2019**, *80*, 144–154. [CrossRef]
16. Pang, B.; Lee, L. Opinion Mining and Sentiment Analysis. *Found. Trends Inf. Retr.* **2008**, *2*, 1–135. [CrossRef]
17. Neidhardt, J.; Rummele, N.; Werthner, H. Can We Predict Your Sentiments by Listening to Your Peers. In *Information and Communication Technologies in Tourism 2016*; Springer: Cham, Switzerland, 2016; pp. 593–603.

18. Garciapablos, A.; Cuadros, M.; Linaza, M.T. Automatic analysis of textual hotel reviews. *Inf. Technol. Tour.* **2016**, *16*, 45–69. [CrossRef]
19. An, H.; Moon, N. Design of recommendation system for tourist spot using sentiment analysis based on CNN-LSTM. *J. Ambient Intell. Humaniz. Comput.* **2019**, 1–11. [CrossRef]
20. Pan, D.; Yuan, J.; Li, L.; Sheng, D. Deep neural network-based classification model for Sentiment Analysis. *arXiv* **2019**, arXiv:1907.02046.
21. Handhika, T.; Fahrurozi, A.; Sari, I.; Lestari, D.P.; Zen, R.I. Hybrid Method for Sentiment Analysis Using Homogeneous Ensemble Classifier. In Proceedings of the 2019 2nd International Conference of Computer and Informatics Engineering (IC2IE), Banyuwangi, Indonesia, 10–11 September 2019; pp. 232–236.
22. Kim, K.; Park, O.; Yun, S.; Yun, H. What makes tourists feel negatively about tourism destinations? Application of hybrid text mining methodology to smart destination management. *Technol. Forecast. Soc. Chang.* **2017**, *123*, 362–369. [CrossRef]
23. Hinterstoisser, S.; Lepetit, V.; Wohlhart, P.; Konolige, K. On Pre-Trained Image Features and Synthetic Images for Deep Learning. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
24. Ghiassi, M.; Lee, S. A domain transferable lexicon set for Twitter sentiment analysis using a supervised machine learning approach. *Expert Syst. Appl.* **2018**, *106*, 197–216. [CrossRef]
25. Kim, K. An improved semi-supervised dimensionality reduction using feature weighting: Application to sentiment analysis. *Expert Syst. Appl.* **2018**, *109*, 49–65. [CrossRef]
26. Ali, F.; Kwak, K.-S.; Kim, Y.-G. Opinion mining based on fuzzy domain ontology and Support Vector Machine: A proposal to automate online review classification. *Appl. Soft Comput.* **2016**, *47*, 235–250. [CrossRef]
27. Parlar, T.; Özel, S.A.; Song, F. QER: A new feature selection method for sentiment analysis. *Hum. Cent. Comput. Inf. Sci.* **2018**, *8*, 10. [CrossRef]
28. Zainuddin, N.; Selamat, A.; Ibrahim, R. Hybrid sentiment classification on twitter aspect-based sentiment analysis. *Appl. Intell.* **2018**, *48*, 1218–1232. [CrossRef]
29. Kumar, A.; Jaiswal, A. Swarm intelligence based optimal feature selection for enhanced predictive sentiment accuracy on twitter. *Multimed. Tools Appl.* **2019**, *78*, 29529–29553. [CrossRef]
30. Pu, X.; Wu, G.; Yuan, C. Exploring overall opinions for document level sentiment classification with structural SVM. *Multimed. Syst.* **2019**, *25*, 21–33. [CrossRef]
31. Adhi, M.S.; Nafan, M.Z.; Usada, E. Pengaruh Semantic Expansion pada Naïve Bayes Classifier untuk Analisis Sentimen Tokoh Masyarakat. *J. RESTI* **2019**, *3*, 141–147. [CrossRef]
32. Fang, C.; Huang, Y. Knowledge-enhanced neural networks for sentiment analysis of Chinese reviews. *Neurocomputing* **2019**, *368*, 51–58.
33. Alowaidi, S.; Saleh, M.; Abulnaja, O. Semantic Sentiment Analysis of Arabic Texts. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 256–262. [CrossRef]
34. Asgarian, E.; Kahani, M.; Sharifi, S. The Impact of Sentiment Features on the Sentiment Polarity Classification in Persian Reviews. *Cogn. Comput.* **2018**, *10*, 117–135. [CrossRef]
35. Agarwal, B.; Mittal, N. Sentiment Analysis Using ConceptNet Ontology and Context Information. In *Prominent Feature Extraction for Sentiment Analysis*; Springer: Cham, Switzerland, 2016; pp. 63–75.
36. Höpken, W.; Fuchs, M.; Menner, T.; Lexhagen, M. Sensing the Online Social Sphere Using a Sentiment Analytical Approach. In *Analytics in Smart Tourism Design: Concepts and Methods*; Xiang, Z., Fesenmaier, D.R., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 129–146. ISBN 978-3-319-44263-1.
37. Akhtar, N.; Zubair, N.; Kumar, A.; Ahmad, T. Aspect based Sentiment Oriented Summarization of Hotel Reviews. *Procedia Comput. Sci.* **2017**, *115*, 563–571. [CrossRef]
38. Ma, E.; Cheng, M.; Hsiao, A. Sentiment analysis – a review and agenda for future research in hospitality contexts. *Int. J. Contemp. Hosp. Manag.* **2018**, *30*, 3287–3308. [CrossRef]
39. Ko, C. Exploring Big Data Applied in the Hotel Guest Experience. *Open Access Libr. J.* **2018**, *5*, 1–17. [CrossRef]
40. Stepchenkova, S.; Kirilenko, A.P.; Li, X. Barriers and Sentiment of the American Tourists Toward Travel to China. In *Tourist Behavior*; Springer: Cham, Switzerland, 2018; pp. 129–139.
41. Bansal, B.; Srivastava, S. Hybrid attribute based sentiment classification of online reviews for consumer intelligence. *Appl. Intell.* **2019**, *49*, 137–149. [CrossRef]
42. Lawani, A.; Reed, M.R.; Mark, T.B.; Zheng, Y. Reviews and Price on Online Platforms: Evidence from Sentiment analysis of Airbnb reviews in Boston. *Reg. Sci. Urban Econ.* **2019**, *75*, 22–34. [CrossRef]

43. Valdivia, A.; Luzón, M.V.; Herrera, F. Sentiment Analysis in TripAdvisor. *IEEE Intell. Syst.* **2017**, *32*, 72–77. [CrossRef]
44. Xiang, Z.; Du, Q.; Ma, Y.; Fan, W. A comparative analysis of major online review platforms: Implications for social media analytics in hospitality and tourism. *Tour. Manag.* **2017**, *58*, 51–65. [CrossRef]
45. Mirzaalian, F.; Halpenny, E. Social media analytics in hospitality and tourism: A systematic literature review and future trends. *J. Hosp. Tour. Technol.* **2019**, *10*, 764–790. [CrossRef]
46. Zhang, W.; Kong, S.; Zhu, Y.; Wang, X. Sentiment classification and computing for online reviews by a hybrid SVM and LSA based approach. *Clust. Comput.* **2019**, *22*, 12619–12632. [CrossRef]
47. Valdivia, A.; Hrabova, E.; Chaturvedi, I.; Luzon, M.V.; Troiano, L.; Cambria, E.; Herrera, F. Inconsistencies on TripAdvisor reviews: A unified index between users and Sentiment Analysis Methods. *Neurocomputing* **2019**, *353*, 3–16. [CrossRef]
48. Schmunk, S.; Hopken, W.; Fuchs, M.; Lexhagen, M. Sentiment Analysis: Extracting Decision-Relevant Knowledge from UGC. In *Information and Communication Technologies in Tourism 2014*; Springer: Cham, Switzerland, 2013; pp. 253–265.
49. Natural Language Toolkit. Available online: http://www.nltk.org/ (accessed on 17 June 2020).
50. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Texts. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; pp. 404–411.
51. Nasar, Z.; Jaffry, S.W.; Malik, M.K. Textual keyword extraction and summarization: State-of-the-art. *Inf. Process. Manag.* **2019**, *56*, 102088. [CrossRef]
52. Campos, R.; Mangaravite, V.; Pasquali, A.; Jorge, A.M.; Nunes, C.; Jatowt, A. YAKE! Keyword extraction from single documents using multiple local features. *Inf. Sci.* **2020**, *509*, 257–289. [CrossRef]
53. Shouzhong, T.; Minlie, H. Mining microblog user interests based on TextRank with TF-IDF factor. *J. China Univ. Posts Telecommun.* **2016**, *23*, 40–46. [CrossRef]
54. Paramonov, I.; Lagutina, K.; Mamedov, E.; Lagutina, N. Thesaurus-Based Method of Increasing Text-via-Keyphrase Graph Connectivity During Keyphrase Extraction for e-Tourism Applications. In Proceedings of the Knowledge Engineering and Semantic Web; Ngonga Ngomo, A.-C., Křemen, P., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 129–141.
55. Gagliardi, I.; Artese, M.T. Semantic Unsupervised Automatic Keyphrases Extraction by Integrating Word Embedding with Clustering Methods. *Multimodal Technol. Interact.* **2020**, *4*, 30. [CrossRef]
56. Ali, F.; El-Sappagh, S.; Riazul Islam, S.M.; Kwak, D.; Ali, A.; Imran, M.; Kyung-Sup, K. A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion. *Inf. Fusion* **2020**, *63*, 208–222. [CrossRef]
57. Cheng, J.; Wang, Z.; Wen, J.; Yan, J.; Chen, Z. Contextual Text Understanding in Distributional Semantic Space. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October 2015; pp. 133–142.
58. Wang, J.; Wang, Z.; Zhang, D.; Yan, J. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, 19–25 August 2017; pp. 2915–2921.
59. Rosso, P.; Yang, D.; Cudremauroux, P. Revisiting Text and Knowledge Graph Joint Embeddings: The Amount of Shared Information Matters! In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019.
60. Wu, W.; Li, H.; Wang, H.; Zhu, K.Q. Probase: A probabilistic taxonomy for text understanding. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, Scottsdale, AZ, USA, 20–24 May 2012; pp. 481–492.
61. Microsoft Concept Graph and Concept Tagging Release. Available online: https://concept.research.microsoft.com/Home/Introduction (accessed on 16 June 2020).
62. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
63. Mikolov, T.; Chen, K.; Corrado, G.S.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
64. Balli, S.; Karasoy, O. Development of content-based SMS classification application by using Word2Vec-based feature extraction. *IET Softw.* **2019**, *13*, 295–304. [CrossRef]

65. Dong, Y.; Liu, P.; Zhu, Z.; Wang, Q.; Zhang, Q. A Fusion Model-Based Label Embedding and Self-Interaction Attention for Text Classification. *IEEE Access* **2020**, *8*, 30548–30559. [CrossRef]
66. Liu, B. *Web Data Mining*; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2007; ISBN 978-3-540-37881-5.
67. Microsoft Knowledge Graph. Available online: https://concept.research.microsoft.com/Home/API (accessed on 17 June 2020).
68. SentiStrength. Available online: http://sentistrength.wlv.ac.uk/ (accessed on 16 June 2020).
69. Chawla, N.V.; Japkowicz, N.; Kotcz, A. Editorial: Special issue on learning from imbalanced data sets. *Sigkdd Explor.* **2004**, *6*, 1–6. [CrossRef]
70. Hu, Q.; Pedrycz, W.; Yu, D.; Lang, J. Selecting Discrete and Continuous Features Based on Neighborhood Decision Error Minimization. *IEEE Trans. Cybern.* **2010**, *40*, 137–150.
71. Lecca, M.; Rizzi, A.; Serapioni, R. GRASS: A Gradient-Based Random Sampling Scheme for Milano Retinex. *IEEE Trans. Image Process.* **2017**, *26*, 2767–2780. [CrossRef] [PubMed]
72. Manning, C.; Schutze, H. *Foundations of Statistical Natural Language Processing*; MIT Press: Cambridge, MA, USA, 1999.
73. Sindhu, I.; Daudpota, S.M.; Badar, K.; Bakhtyar, M.; Baber, J.; Nurunnabi, M. Aspect-Based Opinion Mining on Student's Feedback for Faculty Teaching Performance Evaluation. *IEEE Access* **2019**, *7*, 108729–108741. [CrossRef]
74. Dong, L.; Huang, S.; Wei, F.; Lapata, M.; Zhou, M.; Xu, K. Learning to Generate Product Reviews from Attributes. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; Volume 1, pp. 623–632.

applied sciences

MDPI

Article

Towards the Discovery of Influencers to Follow in Micro-Blogs (Twitter) by Detecting Topics in Posted Messages (Tweets)

Mubashir Ali [1], Anees Baqir [2], Giuseppe Psaila [1,*] and Sayyam Malik [2]

[1] Department of Management, Information and Production Engineering, University of Bergamo, 24129 Bergamo, Italy; mubashir.ali@unibg.it

[2] Faculty of Computing & IT, University of Sialkot, Sialkot 51040, Pakistan; anees.baqir@uskt.edu.pk (A.B.); sayyam.malik@uskt.edu.pk (S.M.)

* Correspondence: giuseppe.psaila@unibg.it; Tel.: +39-035-205-2355

Received: 8 July 2020; Accepted: 12 August 2020; Published: 18 August 2020

Abstract: Micro-blogs, such as Twitter, have become important tools to share opinions and information among users. Messages concerning any topic are daily posted. A message posted by a given user reaches all the users that decided to follow her/him. Some users post many messages, because they aim at being recognized as influencers, typically on specific topics. How a user can discover influencers concerned with her/his interest? Micro-blog apps and web sites lack a functionality to recommend users with influencers, on the basis of the content of posted messages. In this paper, we envision such a scenario and we identify the problem that constitutes the basic brick for developing a recommender of (possibly influencer) users: training a classification model by exploiting messages labeled with topical classes, so as this model can be used to classify unlabeled messages, to let the hidden topic they talk about emerge. Specifically, the paper reports the investigation activity we performed to demonstrate the suitability of our idea. To perform the investigation, we developed an investigation framework that exploits various patterns for extracting features from within messages (labeled with topical classes) in conjunction with the mostly-used classifiers for text classification problems. By means of the investigation framework, we were able to perform a large pool of experiments, that allowed us to evaluate all the combinations of feature patterns with classifiers. By means of a cost-benefit function called "Suitability", that combines accuracy with execution time, we were able to demonstrate that a technique for discovering topics from within messages suitable for the application context is available.

Keywords: social media; micro-blogs (Twitter); towards recommending influencers based on topic classification; investigation framework; comparison of various techniques for topic classification; cost-benefit function

1. Introduction

Micro-blogs have become widely-used online platforms for sharing ideas, political views, emotions and so on. One very famous micro-blog is *Twitter*: it is an online social network that allows users to publish short sentences; every day, millions of messages (also called *tweets*) concerning a very large variety of topics are published (or *posted*) by users. According to [1], Twitter is a famous micro-blogging site where more than 313 million users from all over the world are active monthly.

Due to the importance it has gained, Twitter inspired novel researches concerned with many areas of computer science, in particular data mining [2], sentiment analysis [3], text mining [4], discovering mobility of people [5–7] and so on. For example, tweets are analyzed to find out political friends [8], so this implies that texts are analyzed to detect their political polarity. Another interesting application is detecting communities from networks of users [9], in which sentiment analysis plays an important role; sentiment analysis and opinion mining can be also adopted to study the general sentiment of a given country [10], in order to detect the degree of support to terrorists. We can summarize that most of works concerned with the analysis of tweets are focused on sentiment analysis and opinion extraction; thus, the common perspective is that tweets posted by users are collected and queried to provide useful information about users. We can say that users are analyzed from outside the micro-blog; the results of the analysis are not used to provide a service or a functionality to users of the micro-blog itself.

Nevertheless, many users post a lot of messages, because they wish to influence other users. In fact, when a followed user posts a new tweet, all her/his followers receive it. Typically, users post many messages because they would like to be recognized as influencers in a specific topic. This goal requires a user to have many followers, that are interested in the same topic. Consequently, it is critical, for an influencer, to be interesting for other users and easily found by them. On the contrary, non influencer users would like to easily find interesting influencers to follow.

How to find users to follow? The reasons to decide to follow other users can be various; typically, one reason is affinity of interests: a user would like to follow other users with similar interests. However, currently it is quite hard to find out users that show the same interests, because micro-blog platforms in general (and Twitter in particular) do not provide any end-user functionality or service that recommends users with similar interests; consequently, we are envisioning a new scenario for micro-blog platforms.

This new scenario can become reality only if it is technically possible to realize it. This is the goal of this paper, i.e., addressing the basic problem at the basis of the envisioned functionality: we show that it is effective and efficient to classify messages with topics they talk about. In practice, we demonstrate that it is possible to define a technique that allows for characterizing user interests (in terms of topics) by analyzing their posted messages, that will open the way to build a sort of recommender system that recommends one user with other users having similar interests. At the best of our knowledge, very limited work has appeared in literature concerning this topic.

In this paper, we investigate the definition of an approach based on supervised learning, to discover topics that messages posted by micro-blog users talk about. To this end, we devised an investigation framework whose goal is to apply various combinations of feature patterns (extracted from within posted messages) and classification techniques: this framework has enabled us to identify the best combination to address the problem. In this work, basic and combined n-grams, weighted with a "Term Frequency-Inverse Document Frequency" (TF-IDF)-like metric, are used to extract features from messages to train four of the mostly-used classifiers for text classification, i.e., Naive Bayes (NB), Support Vector Machine (SVM), K- Nearest Neighbors (kNN) and Random Forest (RF). By means of the investigation framework, we performed a comparative analysis of accuracy and execution times; to identify the most suitable solution, we defined a cost-benefit function called *Suitability*, able to balance the benefit of a technique in terms of accuracy with the computational cost of using that technique. We will show that the comparative analysis yielded the solution that we think suitable for discovering topics messages talk about: this is the preliminary step to extend micro-blog user interfaces with functionalities able to suggest influencers to follow. This comparative analysis, that considers both accuracy and execution time, is the distinctive contribution of this paper: in fact, at the best of our knowledge, a similar approach has not been proposed yet.

The rest of the paper is organized as follows. Section 2 gives a brief review of the existing approaches used for text mining applications on micro-blog data sets. Section 3 depicts the envisioned application scenario and defines the specific problem addressed by the paper. Section 4 presents the investigation framework, by discussing the dimensions of the investigation. Section 5 reports about the experimental analysis conducted by means of the investigation framework; by means of results, we perform a comparative analysis of techniques, by considering both their effectiveness (in terms of accuracy) and their computational cost. By means of the cost-benefit function called Suitability, we rank the techniques and we identify the most suitable solution for the application scenario depicted in Section 3. Finally, Section 6 draws the conclusions.

2. Literature Review

To the best of our knowledge, the problem of discovering topics from messages in micro-blogs has not been significantly addressed yet, specifically if the goal is to introduce new functionalities in the micro-blogs interface. Nevertheless, micro-blogs have become precious for many application fields, and many techniques have been developed. In this sense, the related literature is so vast that it is impossible to be exhaustive. In the rest of this section, we propose a brief overview of techniques developed for application areas that are somehow related to our paper.

2.1. Sentiment Analysis and Opinion Mining

Topic discovery is somewhat close to sentiment analysis and opinion mining. Various approaches to perform sentiment analysis and opinion mining on micro-blogs (and Twitter in particular) have been proposed. Their application context is very different with respect to the context and the goal considered in this paper. Nevertheless, it is useful to give an overview of these techniques.

Kanavos et al. [11] proposed an algorithm to exploit the emotions of Twitter users by considering a very large data-set of tweets for sentiment analysis. They proposed a distributed framework to perform sentiment classification. They used Apache Hadoop and Apache Spark to take the benefits of big data technology. They partitioned tweets into three classes, i.e., positive, negative and neutral tweets. The proposed framework is composed of four stages: (i) feature extraction (ii) feature vector construction (iii) distance computation, and (iv) sentiment classification. They utilized hashtags and emoticons as sentiment labels, while they performed classification by adopting the AkNN method (specifically designed for Map-Reduce frameworks).

The study [12] by Hassan et al. evaluated the impact of research articles on individuals, based on the sentiments expressed on them within tweets citing scientific papers. The authors defined three categories of tweets, i.e., positive, negative, and neutral. They observed that articles which were cited in positive or neutral tweets have more impact if compared to articles cited in negative tweets or not cited at all. To perform sentiment analysis, a data-set of 6,482,260 tweets linking to 1,083,535 publications was used.

Twitter data are also very important for companies, so as to exploit them to improve their understanding about the perception by customers of the quality of their products. In [13] authors proposed an approach to process the comments of the customers about a popular food brand, by using tweets from customers. A Binary Tree Classifier was used for discovering the polarity lexicon of English tweets, i.e., positive or negative. To group similar words in tweets, a K-means clustering algorithm was employed.

2.2. Sociological Analysis

The area of sociological analysis is the target of many classification techniques on micro-blog messages.

The paper [14] presents a technique to understand the emotional reactions of supporters of two Super Bowl 50 teams, i.e., Panthers and Broncos. The author applied a lexicon-based text mining approach. About 328,000 tweets were posted during the match by supporters, in which they expressed their emotions regarding different events during the match. For instance, supporters expressed positive emotions when their team scored; on the other hand, they expressed negative emotions when their team conceded a goal. It was concluded that results supported sociological theories of affective disposition and opponent process.

The work [15] shows how the authors used tweets to monitor the opinion of citizens regarding vaccination in Italy, i.e., in favor, not in favor and neutral. For improving the proposed system, different combinations of text representations and classification approaches were used, and the best accuracy was achieved by the combination scheme of bag-of-words, with stemmed n-grams as tokens, and Support Vector Machines (SVM) for classification. The proposed approach fetched and pre-processed tweets related to vaccine and applied SVM to perform classification of tweets and achieved an accuracy of 64.84% , that is acceptable but not very good. The investigation approach is similar to the one adopted in our research, i.e., various combinations of techniques are tested to find the most effective combination.

Geetha et al. [16] aimed to analyze the state of mind expressed on Twitter through emoticons and text in tweets. They developed FPAEC—Future Prediction Architecture Based on Efficient Classification; it incorporates different classification algorithms, including Fisher's linear discriminant classifier, artificial neural networks, Support Vector Machines (SVM), Naive Bayes and balanced iterative reducing; it also incorporates a hierarchical clustering algorithm. In fact, they propose a two-step approach, where clustering follows a preliminary classification step, to aggregate classified data.

2.3. Politics

Politics is an interesting application field of sentiment analysis and opinion mining on micro-blogs. Here, we report a few works.

In [17], the authors proposed a framework to predict the popularity of political parties in Pakistan in 2013 public election, by finding the sentiments of Twitter users. The proposed framework is based on the following steps: (1) collection of tweets; (2) pre-processing of tweets; (3) manual annotation of the corpus. Then, to perform sentiment classification, supervised machine learning techniques such as Naive Bayes (NB), k Nearest Neighbors (kNN), Support Vector Machines (SVM) and Naive Bayes Multinomial (NBMN) were used to categorize the tweets into the predefined labels.

In [18], authors utilized tweets to reveal the views of the leaders of two democratic parties in India. The tweet data-set was collected by using the public twitter accounts, and Opinion Lexicon [19] was used to compute the number of positive, negative and neutral tweets. They proposed a "Twitter Sentiment Analysis" framework, which, after pre-processing of the crawled data-set from Twitter, accumulated opinion lexicon along with classification of tweets into three classes, i.e., positive, negative and neutral, for the evaluation of sentiments of users.

To discover the sentiments of Twitter users, with the aim of exploring their opinions regarding political activities during election days, the authors of [20] proposed a methodology and compared the performance of three sentiment lexicons, i.e., W-WSD, SentiWordNet, TextBlob and two well known machine learning classifiers, i.e., Support Vector Machines (SVM) and Naive Bayes. They achieved better classification results with the W-WSD sentiment lexicon.

In [10], authors utilized tweets to predict the sentiment about Islamic State of Iraq and Syria (ISIS); opinions are organized based on their geographical location. To perform the experimental evaluation, they collected tweets for a period of three days and used Jeffrey Breen's algorithm with data mining algorithms such as Support Vector Machine, Random Forest, Bagging, Decision Trees and Maximum Entropy to classify tweets related to ISIS.

The paper [8] presents a study where the authors exploit tweets to find out political friends. They named their approach a "Politic Ally" which identifies the friends having the same political interest.

2.4. Phishing and Spamming

Aspects related to phishing and spamming can be addressed by analyzing micro-blogs as well, and are close to the problem of topic discovery.

The work [21] proposed an effective security alert mechanism to contrast phishing attacks which targeted users on social networks such as Twitter, Facebook and so on. The proposed methodology is based on a supervised machine learning technique. Eleven critical features in messages were identified: URL length, SSL connection, Hexadecimal, Alexa rank, Age of domain-Year, Equal Digit in host, Host length, Path length, Registrar and Number of dots in host name. Based on these features, messages were classified, to build a classification model able to identify phishing.

Similarly, to deal with spam content being shared on twitter by spammers, Washha et al. [22] introduced a framework called Spam Drift, which combined various classification algorithms, such as Random Forest, Support Vector Machines (SVM) and J48 [23]. In short, they developed an unsupervised framework that dynamically retrains classifiers, used during the on-line classification of new tweets to detect spam.

2.5. Frameworks for Topic Discovery (Interest Mining)

As far as topic discovery (or user interest mining) is concerned, the work [24] proposed a framework for "Tweets Classification, Hashtags Suggestion and Tweet Linking". The framework performs seven activities: (i) data-set selection; (ii) pre-processing of data-set; (iii) separation of hashtags; (iv) finding relevant domain of tweets; (v) suggestion of possible interesting hashtags; (vi) indexing of tweets; (vii) linking of tweets. Thus, topics are represented by hashtags, that are suggested to users. With respect to our approach, discovered topics are very fine grained (at the level of hashtags), because the idea is to suggest hashtags to follow, not users.

In a similar study [25], to detect user interests by automatically categorizing data on the basis of data collected from Twitter and Reddit, authors proposed a methodology comprised of two steps. (i) multi-label text classification model by using Word2vec [26], a predictive model and (ii) topic detection by using Latent Dirichlet Allocation (LDA) [27], a statistical topic detection model based on counting word frequency from a set of documents. A pool of 42,100 documents collected from Redit and manually labeled was used to train the model; then, a pool of 1,573,000 tweets (posted by 1573 users) was used as training set. This work is interesting because it uses Reddit to build the classification model to classify unlabeled tweets from Twitter. However, the scenario is quite different with respect to our paper: in fact, we propose that users wishing to be influencers voluntarily label their posts, with the goal to be recognized as influencers.

The work [28] presents a web-based application to classify tweets into predefined categories of interest. These classes are related to health, music, sport, and technology. The system performs various activities. First of all, they fetch tweets from Twitter and pre-process them; second of all, feature selection from texts is performed; finally, the machine learning algorithm is applied. Although, from a general point of view, it is an interesting system, it is designed to perform analysis of messages from outside the micro-blog. In contrast, our goal is to find out the best technique suitable to discover topics within the micro-blog application.

So, we can say that our envisioned application scenario is quite novel; furthermore, the specific goal of the investigation framework presented in this paper is not to be the end-user solution, but a tool to discover the technique that is most suitable to be executed within the micro-blog application to discover topics.

2.6. Recommendation Techniques

Recommendation techniques have been proposed in the social network world by a multitude of papers. They are so many that it is impossible to report them all. Hereafter, we report those that we consider representative of most recent developments.

The Reference [29] proposed a Recommendation System for Podcast (RSPOD). The system recommends podcasts, i.e., audios, to listen to. The system utilizes the intimacy between social network users, i.e., how well they virtually communicate with each other. RSPOD works (i) by crawling podcast information, (ii) by extracting data from social network services and (iii) by applying a recommendation module for podcasts.

To predict user's rating for several items, [30] considers social trust and user influence. In fact, it is argued that social trust and influence of users can play a vital role to overcome the negative impact on the quality of recommendation caused by sparsity of data. The phenomenon of social trust is based on the sociology theory called "Six Degrees of Separation" [31]: the authors proposed a framework that jointly adopts user similarity, trust and influence, by balancing preferences of users, trust between them and ratings from influential users for recommending shops, hotels and other services. The proposed framework was applied on a data set collected from `dianping.com`, a Chinese platform that allows users to rate the aforementioned services.

According to Chen et al. [32], previous recommendation systems mainly focus on recommendations based on users' preference and overlook the significance of users' attention. Influence of trust relation dwells more on users' attention rather than users' preference. Therefore, an item of a user's interest can be skipped if it does not get his attention. To counter this, they proposed a probabilistic model called Hierarchical Trust-Based Poisson Factorization, which utilizes both users' attention and preferences for social recommendation of movies, music, software, television shows and so on.

Similarly, [33] aimed at accurately predicting users' preferences and relevant products recommendation on social networks by integrating interaction, trust relationships and popularity of products. The key focus of the proposed model is on performing analysis of users' interaction behavior to infer users' latent interaction relationships, based on product ratings and comments. Moreover, the popularity of product is considered as well, to help support decision making for purchasing products.

By emphasizing on the importance of social interaction on recommendation systems [34], presented an approach based on mapping the weighted social interaction for representing interactions among users of a social network, by including historical information about users' behavior. This information is further mined by using an algorithm called Complete Path Mining, which helps find similar social neighbors possessing similar tastes as of the target user. To predict the final ratings of unrated items (such as software, music, movie and so on), the proposed model uses social similar tendencies of the users on complete paths.

To summarize, the reader can see that recommendation techniques are thought to recommend single items (such as posts, podcasts or products) to users, based on the existing relationships among users. Li et al. [35] address the same general problem that we envision in our application scenario, i.e., recommending users to follow: they propose a framework to recommend the 50 users that are more similar to a specific user; they jointly exploit user features (such as ID, gender, region, job, education and so on) and user relationships. In contrast, in our envisioned scenario, we propose a different approach, i.e., recommending other users to establish a relationship with (e.g., to follow) on the basis topics their posts talk about. At the best of our knowledge, this problem has not been addressed yet in literature.

3. Scenario and Problem Statement

In this section, we illustrate the application scenario we are considering, in order to define the problem we address in the rest of the paper.

Suppose a user of a micro-blog platform wants to look for other users to follow, in order to receive their posts. How to find them? Currently, both micro-blog apps and the web sites provide a search functionality to search for users on the basis of a keyword-based search. So, the activity a user has to perform to find out interesting users to follow, that is depicted in the right-hand side of Figure 1 (the block titled *Current Scenario*), can be summarized as follows.

- User $\overline{u}$ performs a keyword-based search, hoping that the specified keywords find out actually interesting users. The search provides the list of users denoted as $R = \langle u_1, \ldots, u_n \rangle$.
- For each user $u_i \in R$ (or for many of them), user $\overline{u}$ opens u_i's profile and looks at it and at messages posted by u_i; if $\overline{u}$ finds that u_i is interesting, $\overline{u}$ asks to follow u_i.

Such a process is quite tedious and boring, so probably user $\overline{u}$ could miss interesting users to follow.

In contrast, we envision a novel functionality for micro-blog apps and web sites: suggesting users based on similar interests. Let us clarify our vision:

- User $\overline{u}$ starts posting some messages, possibly re-posting messages received from followed users.

- At a given time, the application suggests $\overline{u}$ with a list $S = \langle s_1, \ldots, s_m \rangle$ of users potentially having the same interests.
- User $\overline{u}$ looks at the profile of some user $s_j \in S$ and, if $\overline{u}$ finds that s_j is interesting, decides to start following s_j.

Clearly, it is necessary to devise a technique able to learn about user interests. This must necessarily be a multi-label classification technique, that based on the analysis of features extracted from posted messages, builds a model of user interests on the basis of these features.

So, the application scenario we envision, that is illustrated in the left-hand side of Figure 1 (the block titled *Proposed Scenario*), can be described as follows.

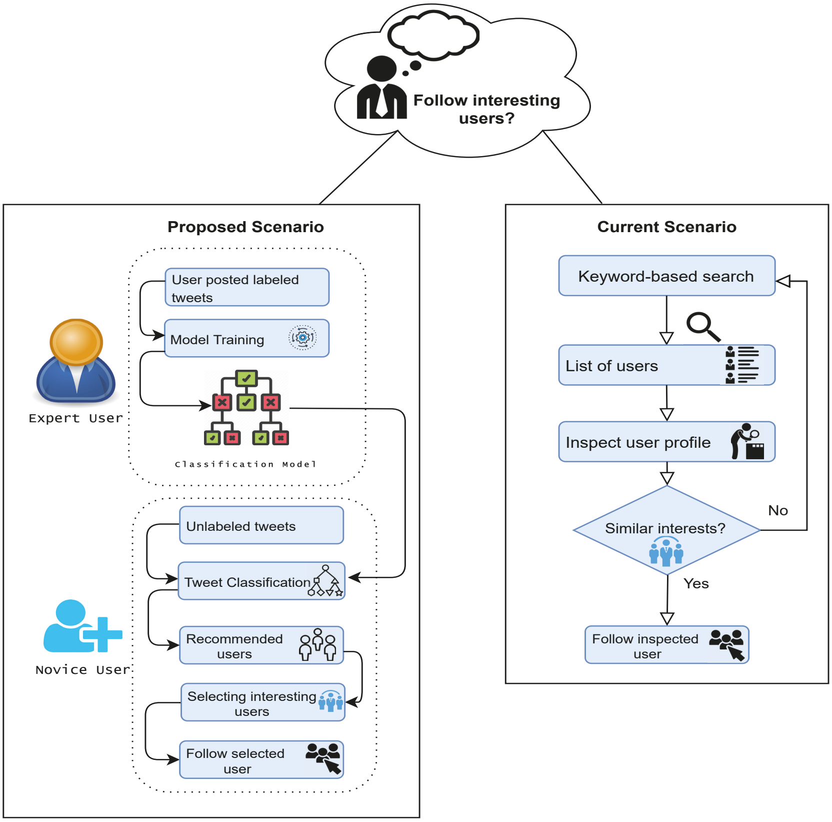


Figure 1. Application scenario.

- Mobile app and web site of a micro-blog should be extended, in order to provide two new functionalities: *Associating Topics to Posted Messages* and *Suggesting Users with Similar Interests*.
- The functionality named *Associating Topics to Posted Messages* should allow users to associate topics to each single post, at the moment they are posting it. A topic will play the role of classification label for the post. This functionality should be not mandatory, and could be appreciated by influencers, i.e., users that are able to (or would like to) influence other users.
- A classification model for topics should be built by analyzing posts that are labeled with topics, on the basis of features extracted from within posted messages.
- The functionality named *Suggesting Users with Similar Interests* applies the classification model to unlabeled messages posted by a user, in order to automatically associate topics to unlabeled

messages. Once the most frequent topics in unlabeled messages posted by user $\overline{u}$ are collected, the application suggests the list $S = \langle s_1, \dots, s_m \rangle$ of users possibly posting messages concerning the same topics of interest for $\overline{u}$.
- User $\overline{u}$ can inspect the profiles of users in S and choose the ones to follow, if any.

In order to avoid misunderstandings, we clearly state that we do not consider two different types of users, i.e., influencers and regular users: any user is equal to other users. However, if a users wishes to be recognized as an influencer, she/he can better succeed if the micro-blog platform provides a tool that helps achieve this aim. In fact, the basic condition for a user to be considered as an influencer is that the number of followers is significantly high; thus, a tool that recommends potential interesting users is the solution. Such a tool could integrate classical text-based search: in fact, we can envision that the micro-blog platform is pro-active in suggesting users; furthermore, text search could be too fine grained to be successful. In other words, we explore the possibility to improve the service provided by micro-blog platforms to users, both those who wish to become influencers and those who wish to find out possibly interesting and emerging influencers.

Clearly, the basic brick to be able to develop the envisioned functionalities is to be able to assign the proper topic to unlabeled messages. The main goal of this paper is to investigate if there exists a classification technique that is suitable for this task, both in terms of effectiveness and in terms of efficiency. The specific problem that must be addressed by the wished technique is defined as follows.

Problem 1. *Consider a set $LP = \{lp_1, \dots, lp_n\}$ of labeled posts; each $lp_i = \langle mt_i, at_i \rangle$ denotes a labeled post, where mt_i is the message text and at_i is the assigned topic.*

Consider a second set $UM = \{umt_1, \dots, umt_m\}$ of unlabeled messages umt_j. Based on the set of labeled posts LP, a classification model $C(umt)$ must be built, such that given a message text $umt_j \in UM$, $C(umt_j) = tp_j$, i.e., the classification model C provides the topic tp_j of the umt_j message.

In the rest of the paper, we will address Problem 1, looking for the technique based on text classification that provides the best compromise between accuracy (as far as topic detection is concerned) and efficiency. In fact, if we are able to demonstrate that there exists a technique suitable to solve Problem 1, the way to further investigate how to rank influencers to suggest to users can be taken.

4. The Investigation Framework

In this section, we introduce the framework we built to investigate how to discover topics messages talk about, as reported in Problem 1. First of all, we discuss the dimensions of investigation we considered (Section 4.1); then, we present technical aspects of the framework in details.

4.1. Dimensions of the Investigation

Problem 1 is a multi-label text classification problem. Thus, through the investigation framework, two dimensions must be investigated.

- *Feature extraction.* Message texts must be represented by means of a pool of features, that denote texts at a level of granularity that makes the classifier effective. However, many patterns of features can be adopted to represent texts: so effectiveness and efficiency of classifiers are significantly affected by the specific feature pattern adopted to represent texts.
- *Classification technique.* Different classification techniques behave differently, so it is necessary to evaluate the behavior of a pool of classification techniques.

Figure 2 graphically reports the dimensions of the investigation: the reader can see that the two dimensions are orthogonal. Thus, the goal of the investigation framework is to experiment all combinations, in order to find the best one to solve Problem 1. Hereafter, we separately discuss each dimension.

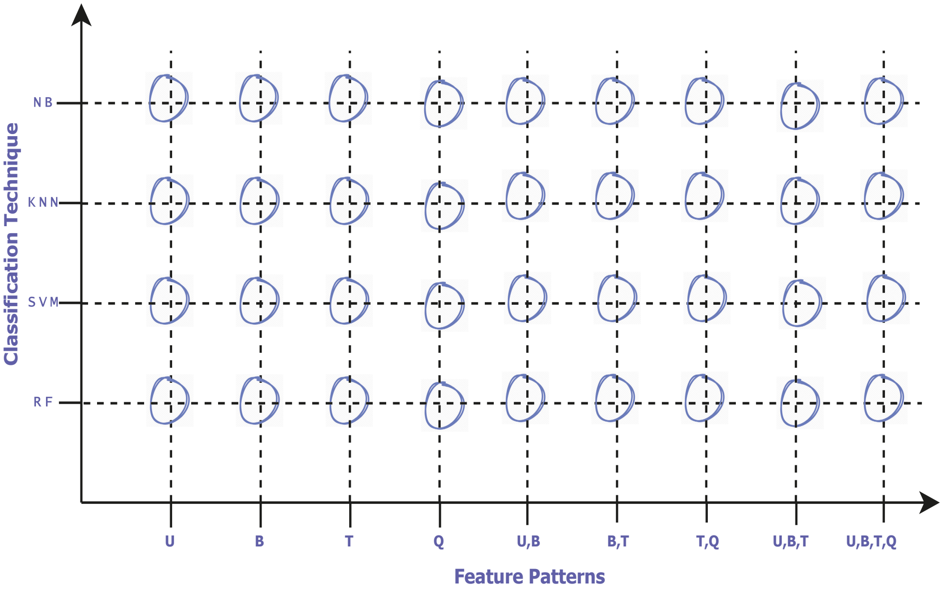


Figure 2. Dimensions of the investigation.

4.1.1. Feature Extraction

In order to apply the classification technique, we need to extract features to classify from texts, in order to obtain a different representation of texts. We decided to adopt the *n-gram model*, that is widely adopted in text classification.

Hereafter, we shortly introduce the four basic n-gram patterns we adopted in our investigation.

- **Uni-gram patterns.** In our model, a *uni-gram* is a single word (or token) that is present in the text. Uni-gram patterns are singleton patterns, i.e., a single word is a pattern itself (i.e., n-grams with $n = 1$).
  Uni-gram patterns do not consider the relative position of words in the text.
- **Bi-gram patterns.** A *bi-gram* is a sequence of two consecutive uni-grams (n-grams with $n = 2$), i.e., two consecutive words in the text.
- **Tri-gram patterns.** A *tri-gram* is a sequence of three consecutive uni-grams (n-grams with $n = 3$), i.e., three consecutive words in the text.
- **Quad-gram patterns.** A *quad-gram* is a sequence of four consecutive uni-grams (n-grams with $n = 4$), i.e., four consecutive words in the text.

With these premises, we can represent a document d (a message text, in our context) as a vector of terms, i.e., $d[j]$ is the j-th term in the document. When we consider n-grams, the document is represented as a vector of n-grams, i.e., $d[j]$ is the n-gram whose first word is in position j in the original document (of course, if $n = 1$, the vector of uni-grams and the vector of words coincide.).

Table 1 reports four different ways of representing a sample document, based on uni-grams, bi-grams, tri-grams and quad-grams, by reporting the different vectors that represent the same document. For example, if we consider the case $n = 3$ in Table 1, d contains only two items, i.e., $d[1]$ and $d[2]$.

Table 1. An example of n-gram patterns for the string `"this is a sentence"`.

N-Grams	Feature Vectors
Uni-grams: $n = 1$	$d[1]$ ="this", $d[2]$ ="is", $d[3]$ ="a", $d[4]$ ="sentence"
Bi-grams: $n = 2$	$d[1]$ ="this is", $d[2]$ ="is a", $d[3]$ ="a sentence"
Tri-grams: $n = 3$	$d[1]$ ="This is a", $d[2]$ =is a sentence"
Quad-grams: $n = 4$	$d[1]$ ="This is a sentence"

Moving from the methodology proposed in [36], we consider also combined features, i.e., features obtained by combining basic features (i.e., uni-grams, bi-grams, tri-grams and quad-grams).

Given z sets of basic features BF_i, with $1 \leq i \leq z$, a set CF of complex features is obtained by means of the Cartesian product of sets BF_i, i.e., $CF = BF_1 \times BF_2 \times \cdots \times BF_z$. Thus, a feature in CF is a tuple of z basic features (n-grams).

As an example, consider Table 1. A feature obtained by combining a uni-gram and a bi-gram is the tuple ⟨"this","a sentence"⟩.

In our framework, we considered the four basic feature patterns and five complex feature patterns. In Table 2, we report them and the corresponding abbreviation we will use throughout the paper.

Table 2. Basic and complex feature patterns computed by the investigation framework.

Abbreviation	Pattern
U	Uni-grams
B	Bi-grams
T	Tri-grams
Q	Quad-grams
U,B	Product of uni-grams and bi-grams
B,T	Product of bi-grams and tri-grams
T,Q	Product of tri-grams and quad-grams
U,B,T	Product of uni-grams, bi-grams and tri-grams
U,B,T,Q	Product of uni-grams, bi-grams, tri-grams and quad-grams

Feature weight. In order to help the construction of the classification model, features are weighted. Typically, in text classification the most-frequently adopted metric is *Term Frequency-Inverse Document Frequency* (TF-IDF) [37]. It is a numerical score which denotes the importance of a term in a collection of documents. TF-IDF is the combination of two scores which are called *Term Frequency* and *Inverse Document Frequency*. The comparative analysis in [38] demonstrated that TF-IDF significantly improves the effectiveness of classifiers.

The score balances the importance of a term for a given document with respect to its capability of characterizing a small number of documents. The rationale is that if a term is highly frequent in the collection, it does not characterize a subset of documents; thus, terms that appear in many documents cannot be considered relevant features for any document.

Consider a set $D = \{d_1, \ldots .d_n\}$ of documents, where each document $d_i \in D$ is a vector of terms (in the broadest sense, i.e., terms can be either n-grams or tuples of n-grams). The Term Frequency $Tf(t,d)$ of a term t in a document d is the number of times t appears within d on the total number of terms in d (see [39]). It is defined as:

$$Tf(t,d) = \frac{|\{j|d[j] = t\}|}{|d|}.$$

The Inverse Document Frequency $Idf(t,D)$ of a term t in the collection (of documents) D measures the capability of t of denoting a small set of documents in D: the lower the number of documents in which t appears, the greater its Idf score [39]. It is defined as:

$$Idf(t, D) = log_e \frac{|D|}{|\{d \in D | (\exists j | d[j] = t)\}|}.$$

By combining Tf and Idf, we obtain the overall $TfIdf(t, d, D)$ score of a term t within a document d belonging to a collection of documents D, as follows:

$$TfIdf(t, d, D) = Tf(t, d) \times Idf(t, D).$$

In our model, terms are either basic n-grams (basic feature patterns denoted as **U**, **B**, **T** and **Q**), or combined n-grams, such as **U,B** and so on (see Table 2): thus, we apply the $TfIdf$ metric to rank these features. However, we do not compute TF-IDF on a document basis, but on a class basis: the frequency of a term is the number of documents in the class that contain the term; the inverse document frequency should be properly called *Inverse Class Frequency*, because we count the number of classes that contain the term on the total number of classes. Formula 1 formally defines the weight.

$$Weight(t, c, C) = \frac{|\{d | (d.class = c \wedge \exists j | (d[j] = t))\}|}{|\{d | d.class = c\}|} \times log_e \frac{|C|}{|\{c_i \in C | (\exists j | d[j] = t \wedge d.class = c_i)\}|} \quad (1)$$

in other words, the weight of a term t in a class $c \in C$ is the frequency of t within the documents that belong to that class, multiplied by the inverse frequency of t among all classes in C. Notice that with $d.class$ we denote the class which document d belongs to.

In Section 5.2, we perform experiments with the full set of features and with the strongest 80%, 65% and 50% features, on the basis of function $Weight(t, c, C)$ defined in Formula 1.

4.1.2. Classification Techniques

The second dimension of investigation is to find out the classification technique that demonstrates to be more suitable for the application scenario. Recall from Problem 1 that the classifier has to discover the topic tp_j that an unlabeled message umt_j talks about. Hereafter, we briefly introduce the four classification techniques we considered in our investigation framework.

- **Naive Bayes (NB).** The Naive Bayes classifier [40] is a simple, fast, efficient, easy to implement and popular classification technique for texts: in fact, this technique is quite efficient as far as computation time is concerned; however, it performs well when features behave as statistically independent variables.
  In short, it is a probabilistic classification technique, which completely depends on the probabilistic value of features. For each single feature, the probability that it falls into a given class is calculated. It is widely used to address many different problems, such as for predicting social events, for denoting personality traits, for analyzing social crime, and so on.
- **Support Vector Machine (SVM).** Support Vector Machine classifiers are widely used for classification of short texts. This classification technique is based on the principle of structured risk minimization [41]: given the hyper-space of features, in which each point represents a document, it creates a hyper-plane h that divides the data into two sets (i.e., the hyper-space is divided into two semi-spaces by the hyper-place); the algorithm tries to identify the hyper-plane that maximizes the distance from each point (the distance is called *margin*) because the greater the margin, the lower the risk that a point falls into the wrong semi-space. In the test phase, a data point is categorized depending on the semi-space it falls into. The technique has been extended and adapted to support multi-label classification [42].
- **K-Nearest Neighbors (kNN).** K-Nearest Neighbors (kNN) classifiers are widely used for classification of short texts. During the test phase, given a new document $\overline{d}$, the distance between document $\overline{d}$ and all the documents in the training corpus is measured, by employing a similarity or a dissimilarity measure. Then, the set $N(\overline{d})$ that contains the K nearest neighbors among the entire training corpus is obtained; the class label having the largest number of documents in $N(\overline{d})$

becomes the class of $\overline{d}$ [43]. For example, if $N(\overline{d})$ contains 15 nearest neighbors to document $\overline{d}$, where seven documents are labeled with the `"politics"` class, four documents are labeled with the `"sports"` class, three documents are labeled with the `"weather"` class and one document is labeled with the `"health"` class, then $\overline{d}$ is labeled with the `"politics"` class.

- **Random Forest (RF).** It is a known supervised learning method for classification devised by Ho [44]. It is an evolution of classical tree-based classifiers. The name of the technique is motivated by the fact that, during the training phase, many classification trees are generated: we can say that the classification model is a *forest* of classification trees. During the test phase, all the classification tress are independently used to classify the unclassified case: the class assigned by the majority of trees is chosen as class label assigned to the unclassified case.
  This technique is very general and widely used in many application contexts, not only for text classification [45].

4.2. Framework Overview

The investigation framework is composed of many modules. First of all, we give a high-level overview of them, describing the task performed by each single module.

- **Module M1-Pre-processor.** The module named *Pre-processor* performs many pre-processing activities on the data set, i.e., the set of labeled messages that constitutes the input data set for the investigation framework. Specifically, it performs tokenization, stop-word removal, special symbol elimination, and stemming (that are typical pre-processing techniques adopted in information retrieval). Specifically, stemming is important to reduce dimensionality of features: in fact, natural languages provide many different forms for the same word (for instance, singular and plural); stemming reduces words to the root form.
  The result of the *Pre-processor* module is the corpus of messages, where each document is represented as an array of stems.
- **Module M2-Feature Extractor.** After the *Pre-processor* is executed, the module named *Feature Extractor* extracts features that represent messages. In this module, messages are translated into vectors of basic and combined feature patterns, as reported in Table 2. The $Weight(t, c, C)$ score defined in Formula 1 is computed for each term (i.e., feature) belonging to the training set.
- **Module M3-Multi-Classifier.** The final module is called *Multi-Classifier*, because it has to apply all the four different classification techniques we discussed in Section 4.1.2. In fact, the goal of the investigation framework is to understand which is the best combination of feature pattern and classification technique (recall Figure 2). Thus, for each feature pattern, the module builds four classification models and tests them to evaluate the accuracy of classification.

Figure 3 reports the organization of the framework, by illustrating data flows between and inside modules. They are discussed in details hereafter.

The framework is implemented in the Python programming languages, by exploiting the libraries `nltk` for pre-processing, `sklearn` for feature extraction, generation of n-gram combinations, training and testing of classifiers.

4.2.1. External Module EM1-Message Collector

This module is responsible to gather messages from the source micro-blog (in our case, Twitter) and support researchers to label messages with class labels denoting topics.

Since our investigation framework is designed to be independent of the specific source micro-blog, we decided to consider it as an external module that can be replaced with a different one, suitable to gather data from a different micro-blog.

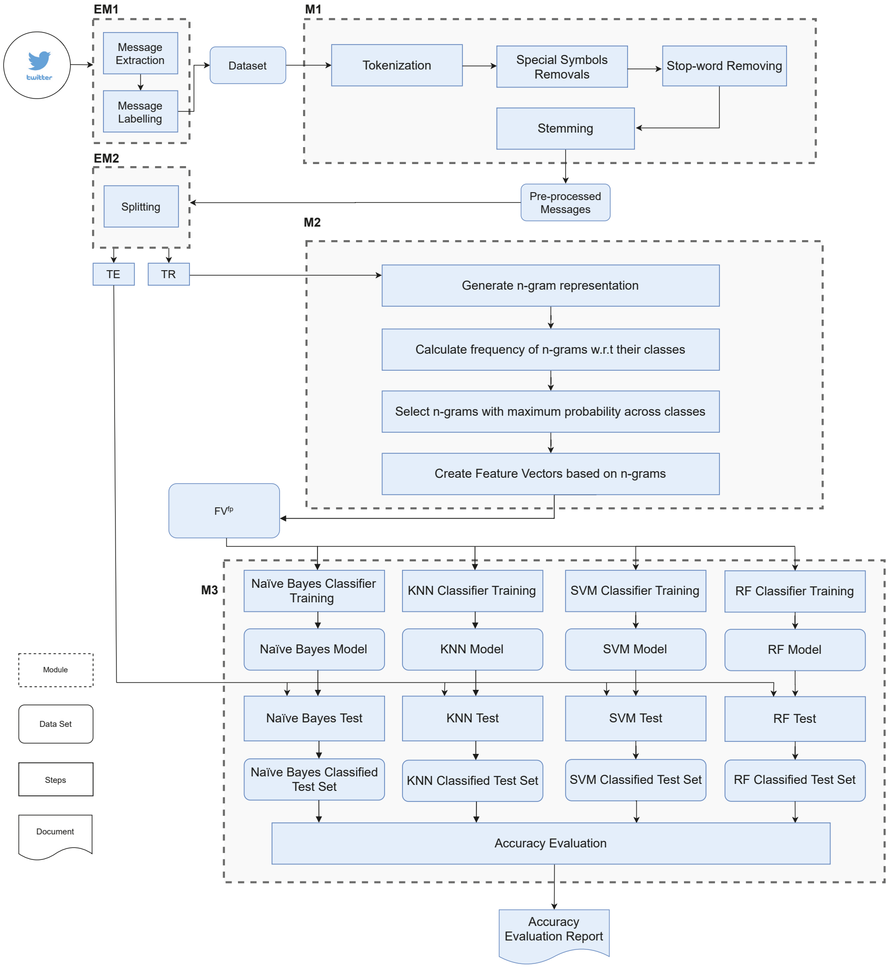


Figure 3. The Investigation Framework.

4.2.2. Module M1-Pre-Processor in Details

When users write messages, they write punctuation, single characters and stop-words that are not useful for topic classification (and even decrease the accuracy of classification). So, before features are extracted, messages must be pre-processed in order to be cleaned from noise. Specifically, module *Pre-processor* performs text tokenization, special symbol removal, stop-words filtering and stemming. Hereafter, we describe these activities in more details.

Let us denote the input data set as $T = \{lp_1, lp_2, \ldots, \}$, where each lp_i is a labeled message, such that $lp_i = \langle mt_i, at_i \rangle$, i.e., mt_i is the message text and at_i is the label class or topic associated to the message.

For each message $lp \in T$, on its message text $lp.mt$ the module performs the following processing steps, in order to generate the set T_p of pre-processed messages.

1. The $lp.mt$ message is tokenized, in order to represent it as a vector d_1 of terms, where a term is a token found within the message text.

2. From vector d_1, we obtain vector d_2 by removing special symbols, punctuation marks, numbers and special characters.
3. Stop-word removal is performed, by comparing each term in d_2 with *NLTK* [46], a static list of stop-words. We formalize this process by defining the recursive function *RemoveSW* hereafter.

$$RemoveSW(d, pos) = \\ = \begin{cases} RemoveSW(remove(d, pos), pos) & \text{if } (1 \leq pos \leq |d|) \wedge (|d[pos]| \leq 2 \vee d[pos] \in S_L)) \\ RemoveSW(d, pos+1) & \text{if } (1 \leq pos \leq |d|) \wedge \neg(|d[pos]| \leq 2 \vee d[pos] \in S_L)) \\ d & \text{if } \neg(1 \leq pos \leq |d|) \end{cases}$$

where d is the message represented as a vector of terms, $|d|$ is the size of the vector, pos denotes a position index, $d[pos]$ denotes the term (string within vector d in position pos), $|d[pos]|$ denotes the length of the term (string) in position pos in vector d. Furthermore, S_L is the list of stop-words, while function $remove(d, pos)$ removes the item in position pos from vector d. The function is defined by the following formula.

$$remove(d, pos) = \begin{cases} d[1, (pos-1)] \bullet d[(pos+1), |d|] & \text{if } 1 < pos < \ |d| \\ d[2, |d|] & \text{if } pos = 1 \\ d[1, (pos-1)] & \text{if } pos = |d| \end{cases}$$

where with $d[i, j]$ we denote the sub-vector with items from position i to position j and the $\bullet$ denotes an operator that concatenates two vectors.

The d_3 vector representing the message without stop-words is obtained by calling the *RemoveSW* function as $d_3 = RemoveSW(d_2, 1)$.
4. After stop-word removal, stemming is performed on vector d_3 by applying the *Porter stemming algorithm* [47]; we obtain the final d_4 vector, i.e., the vector of terms that represent the $lp.mt$ message text after pre-processing. The d_4 vector is paired with the class label $lp.at$, obtaining the pair $lm_p = \langle d_4, lp.at \rangle$ that is inserted into T_p, the set of pre-processed messages.

T_p is the final output of this module: the source data set T has been transformed into T_p, where instead of strings, message texts are represented by vectors of terms.

4.2.3. External Module EM2-Data Splitter

The pre-processed data set T_p is now split into training set TR and test set TE. The training set becomes the input of module *M2-Feature Extractor*, while the test set TE will be used by module *M3-Multi-classifier*, for computing the accuracy. Notice that TE contains labeled messages: this is necessary for validating classification and compute the accuracy, by computing true positives, false positives, true negatives and false negatives simply by comparing the topic assigned by the classifier to the message and the label originally associated with the message.

This is an external module of the investigation framework, if compared to modules M1, M2 and M3, that are the core modules of the framework. This choice is motivated by the need for flexibility. In fact, different techniques for splitting could be used; this way, the investigation framework is parametric with respect to data splitting.

4.2.4. Module M2-Feature Extractor in Details

Module *M2-Feature Extractor* receives the training set TR extracted from the overall set of pre-processed messages T_p. Its goal is to give different representations of each labeled message $lm_p \in TR$, based on the basic and combined feature patterns reported in Table 2.

The module generates nine different versions of the training set TR, one for each feature pattern, denoted as TR^U, TR^B, TR^T, TR^Q, $TR^{U,B}$, $TR^{B,T}$, $TR^{T,Q}$, $TR^{U,B,T}$ and $TR^{U,B,T,Q}$. These are intermediate results, necessary to generate the actual output of the module, i.e., a pool of feature vectors FV^{fp} (where fp denotes the feature pattern, as in Table 2): each FV^{fp} contains a feature vector for each topical class. Let us start by describing the generation of TR^{fp}.

- First, $TR^U = TR$, because uni-grams coincide with single terms in vectors $lm_p.d_4$ representing message texts (i.e., $lm^U.d = lm_p.d_4$).
- Each $lm^B \in TR^B$ derives from the corresponding basic version $lm_p \in TR$, where $lm^B.d$ is a vector of bi-grams.
  Similarly, training sets TR^T and TR^Q contains descriptions lm^T and lm^Q of messages whose vectors $lm^T.d$ and $lm^Q.d$ are vectors of tri-grams and quad-grams, respectively.
- Training sets based on combined feature patterns are derived from training sets based on basic patterns.
  Given a message lm_p, its representations based on combined feature patterns are obtained as follows:
  - for $lm^{U,B} \in TR^{U,B}$, $lm^{U,B}.d = lm^U.d \times lm^B.d$;
  - for $lm^{B,T} \in TR^{B,T}$, $lm^{B,T}.d = lm^B.d \times lm^T.d$;
  - for $lm^{T,Q} \in TR^{T,Q}$, $lm^{T,Q}.d = lm^T.d \times lm^Q.d$;
  - for $lm^{U,B,T} \in TR^{U,B,T}$, $lm^{U,B,T}.d = lm^U.d \times lm^B.d \times lm^T.d$;
  - for $lm^{U,B,T,Q} \in TR^{U,B,T,Q}$, $lm^{U,B,T,Q}.d = lm^U.d \times lm^B.d \times lm^T.d \times lm^Q.d$.

Once the training sets are prepared, for each of them (that generically we will denote as TR^{fp}) the module performs the following activities.

1. For each message $lm_i^{fp} \in TR^{fp}$, a set of terms s_i^{fp} is derived from the vector of terms $lm_i^{fp}.d$:
$$s_i^{fp} = \{t | \exists pos(1 \leq pos \leq |lm_i^{fp}.d| \wedge lm_i^{fp}.d[pos] = t)\}$$
   so that duplicate occurrences of a term in $lm_i^{fp}.d$ becomes a unique occurrence in s_i^{fp}.
2. The frequency matrix $Freq^{fp}[t,c]$ is built, where t is a term and c is a class (or topic). To obtain the $Freq^{fp}$ matrix, first of all the module builds tc_i^{fp}, a set of (term, class, message identifier) triples (t,c,i) obtained as
$$tc_i^{fp} = s_i^{fp} \times \{lm_i^{fp}.at\} \times \{i\}$$
   that is, by performing the Cartesian product among the set s_i^{fp} of terms in the message, the singleton set of the class label (topic) $lm_i^{fp}.at$ associated to the message and the singleton set containing i (i.e., the identifier of the message). All the tc_i^{fp} sets are united into the TC^{fp} set, i.e., $TC^{fp} = \cup_{\forall lm_i^{fp} \in TR^{fp}}(tc_i^{fp})$.
   Each single item of the $Freq^{fp}$ matrix is then computed as follows:
$$Freq^{fp}[t,c] = |\{(t_j, c_J, i_j) \in TC^{fp} | t_j = t \wedge c_j = c\}|$$
   where we count, for each term t and each class c, the number of different documents, associated to class c, which term t occurs in (the third element i_j in triples is necessary to distinguish term occurrences coming from different messages).
3. For each term t in each class c, the module computes the $Weight(t,c,C)$ score (defined in Formula 1), where C is the set of all class labels. We denote the weight for the feature pattern fp as w^{fp}; it is defined as:
$$W^{fp}(t,c) = Weight(t,c,C) = \frac{Freq^{fp}[t,c]}{\sum_{\forall t_j} Freq^{fp}(t_j,c)} \times log_e\left(\frac{|C|}{|\{c_k | Freq^{fp}[t,c_k] > 0\}|}\right) \quad (2)$$
   where C is the overall set of class labels. In the product on the right-hand side of the formula, the first operand is the term frequency, while the second operand is the inverse document frequency.

4. Finally, for each class $c_k \in C$, the feature vector $f^{fp}(c_k)$ for the given feature pattern is built, where each item is a pair (t, w), where t is the term and $w = w(t, c_k)$ is the weight.
   The sets FV^{fp} of feature vectors $f^{fp}(c_k)$, where fp denotes the feature pattern (as in Table 2), are the final output of module M2.

4.2.5. Module M3-Multi-Classifier in Details

Module *M3-Multi-classifier* performs the last step of the investigation process, i.e., it builds the classification models by training the classifiers, then exploits the classification models to label the test set. It is called *multi-classifier* because it uses all the four classification techniques shortly presented in Section 4.1.2, to train a classification model and label the test set.

Let us describe the process performed by Module M3 in details. The module receives two inputs: the pool of feature vector sets FV^U, FV^B, FV^T, FV^Q, $FV^{U,B}$, $FV^{B,T}$, $FV^{T,Q}$, $FV^{U,B,T}$ and $FV^{U,B,T,Q}$, generated by module M2, and the test set TE, generated by the external module EM2. For each one of the training sets and for each one of the classification techniques, the module performs the following activities.

- The specific classification technique (Naive Bayes, SVM, kNN and Random Forest) is applied, to obtain a classification model $cm^{fp,ct}$ (where fp is the feature pattern and ct is the classification technique) for each feature pattern, using the corresponding FV^{fp} as training set.
- For each classification model $cm^{fp,ct}$, the messages in the test set TE are labeled accordingly. The classified test sets so far obtained contain both the labels provided by the classifier and the labels assigned by humans that prepared the overall data set.

At this point, the module performs the accuracy evaluation, i.e, it evaluates accuracy of classification for all the classified test sets, in order to produce a final report, that is the outcome of the investigation framework.

5. Experimental Evaluation

The investigation framework was run on a data set specifically collected. In Section 5.1, we present both the way we collected and prepared the data set, as well as the metrics we adopted to evaluate the classification results. In Section 5.2, we present the experiments and discuss the results, as far as the effectiveness of classification is concerned, while in Section 5.3 we present the sensitivity analysis of classifiers.. Then, Section 5.4 considers execution times and introduces the metric called *Suitability*.

5.1. Data Preparation and Evaluation Metrics

To perform the experimental evaluation through the proposed investigation framework, we performed data collection and labeling. Data collection is the process of collecting messages (from Twitter) that are relevant to the problem domain. It is a crucial step, because it strongly determines the results obtained by classifiers. Messages were collected from Twitter by using Tweepy API [48]; 133,306 messages were collected from different accounts.

The next step was to manually label the messages with a pool of predefined topics. In this process, we involved five volunteer students of the Masters degree at University of Sialkot (Pakistan), to label messages. Each message was labeled by two different students, that worked separately: in the case two different labels were assigned to the same message, the message was discarded from the data set. This way, only messages labeled with the same class by two different students were considered: messages that did not clearly talk about one of the selected topics were not considered.

Hereafter, we list the topics considered as classes and the criteria adopted for labeling messages with each single topic.

- *Business*: Messages talk about stocks, business activities, oil prices, Wall Street and companies' shares.

- *Health*: Messages that talk about disease, medicine, surgeries, viruses, hospitals and related arguments are included in this topic.
- *Politics*: Messages regarding elections, democracy, government and its policies are included in this topic.
- *Entertainment*: Messages regarding show business, movies, music, TV shows and similar arguments belong to this topic.
- *Sports*: Messages regarding all kinds of sports, athletes, matches and sports tournaments belong to this topic.
- *Technology*: Messages talk about new tech, gadgets, software and related information.
- *Weather*: Messages talk about weather, rains, storms and weather forecasting.

The list of topics was inspired by [49], that proposed a list of categories for classifying sensitive tweets; we did not consider all the list proposed in [49], because some of the proposed categories did not denote topics that users would use to label messages (e.g., racism); we selected and integrated those that, presumably, could be often used by users. Table 3 provides a sample message for each one of the topical classes.

Table 3. Chosen topics with example messages.

Topics	Example Messages
Business	Strong growth and rare profits make Veeva's stock worth the sky-high valuation Profit at the Canadian bank's U.S. retail segment rose 13% in its latest quarter.
Health	CDC recommends that boys and girls get vaccinated for HPV between 11 & 12 years of age Obesity now affects 17% of all children and adolescents in the U.S
Politics	Joe Biden holds town hall event in Greenville, SC Britain faces no deal on Brexit as Boris Johnson handles this crisis
Entertainment	'Ford v Ferrari', 'Uncut Gems', 'Judy' Headed to Telluride Film Festival Dwayne Johnson is returning to 'WWE Smackdown' for Fox launch
Sports	Uganda's Nakaayi wins the women's 800 metres final in 1:58.04 It's Man Utd's worst start to a league season in 30 years
Technology	Gatwick Airport commits to facial recognition tech at boarding Facebook to create VR world called Horizon
Weather	Tropical Storm 'Narda' made landfall near Lazaro Cardenas, Mexico Feisty thunderstorms have formed and are tracking through the Dakotas tonight

In order to have a homogeneous distribution among classes, the training set TR contained 3500 messages for each class, while the test set TE contained 1500 messages for each class. Consequently, the training set TR contained 24,500 messages, while the test set TE contained 10,500 messages; the total number of messages was 35,000, that constitute the input for the investigation framework. All messages were written in English.

Remember that messages in the test set TE were labeled by hand as well, in order to allow module M3 to automatically compute accuracy.

To evaluate the results, the investigation framework computed accuracy, precision, recall and F1-measure. These measures are typical metrics adopted in information retrieval. Since we operated in a context of multi-label classification, we adopted the definitions reported in [50,51].

Given a set $C = \{c_1, c_2, \ldots, c_n\}$ of class labels, for each class c_j we define the following counts:

- TP_j (true positives) is the number of items correctly assigned to class c_j;
- FP_j (false positives) is the number of items incorrectly assigned to class c_j;
- FN_j (false negatives) is the number of items incorrectly not assigned to class c_j;
- TN_j (true negatives) is the number of items correctly not assigned to class c_j.

For each class c_j, we can define the four above-mentioned metrics.

- $Accuracy_j$ is the number of messages properly associated and properly not associated with class c_j on the total number of messages, i.e., $Accuracy_j = \frac{TP_j+TN_j}{TP_j+TN_j+FP_j+FN_j}$.
- $Precision_j$ is the fraction of messages correctly labeled with class c_j on the total number of messages labeled with c_j by the classifier, i.e., $Precision_j = \frac{TP_j}{TP_j+FP_j}$.
- $Recall_j$ is the fraction of messages properly labeled with class c_j on the total number of messages that have to be labeled with c_j, i.e., $Recall_j = \frac{TP_j}{TP_j+FN_j}$.
- $F1\text{-}measure_j$ is a synthetic measure that combines precision and recall, i.e., $F1\text{-}measure_j = \frac{Precision_j \times Recall_j}{Precision_j+Recall_j} \times 2$.

Since we are in a context of multi-label classification, we have to compute a general global version of each measure. This is usually done by averaging the values computed for each class. Consequently, $Accuracy = (\sum_{c_j \in C} Accuracy_j)/|C|$, $Precision = (\sum_{c_j \in C} Precision_j)/|C|$, $Recall = (\sum_{c_j \in C} Recall_j)/|C|$, $F1\text{-}measure = (\sum_{c_j \in C} F1\text{-}measure_j)/|C|$.

We are now ready to discuss the results of our investigation, based on the two dimensions discussed in Section 4.1.

5.2. Experiments and Comparison of Classifiers

Based on the dimensions of investigation discussed in Section 4.1, we performed a large number of experiments, that involved the four classification techniques presented in Section 4.1.2.

Let us start considering the results obtained by the Naive Bayes classifier. Table 4 is organized as follows: for each basic n-gram pattern, i.e., **U**, **B**, **T** and **Q**, as well as for each combined feature pattern **U,B**, **B,T**, **T,Q**, **U,B,T** and **U,B,T,Q**, the full set of features (100%) and the most relevant 80%, 65% and 50% of features, on the basis of their weight defined in Formula 1 are used to perform experiments.

Table 4. Experimental results for Naïve Bayes classifier.

Feature Pattern	No. of Features	Accuracy	Precision	Recall	F1-Measure
U	24,961 (100%)	83.88	84.39	83.88	83.86
	19,969 (80%)	83.93	84.39	83.93	83.91
	16,225 (65%)	83.99	84.41	83.99	83.96
	12,481 (50%)	83.79	84.16	83.79	83.76
B	165,704 (100%)	70.36	76.82	70.36	70.82
	132,563 (80%)	68.62	76.48	68.62	69.23
	107,708 (65%)	67.24	75.87	67.24	67.92
	82,852 (50%)	65.92	75.45	65.92	66.75
T	181,514 (100%)	44.28	81.2	44.28	46.92
	145,211 (80%)	42.73	81.09	42.73	45.21
	117,984 (65%)	41.61	80.94	41.61	43.9
	90,757 (50%)	40.45	80.89	40.45	42.6
Q	168,482 (100%)	30.68	84.46	30.68	30.94
	134,786 (80%)	29.86	84.82	29.86	29.78
	109,513 (65%)	29.1	84.76	29.1	28.77
	84,241 (50%)	28.48	84.83	28.48	27.92
U,B	190,665 (100%)	84.3	85.07	84.3	84.28
	152,532 (80%)	84.48	85.21	84.48	84.46
	123,932 (65%)	84.35	85.06	84.35	84.33
	95,333 (50%)	84.53	85.15	84.53	84.51
B,T	347,218 (100%)	70.42	76.87	70.42	70.88
	277,774 (80%)	68.73	76.41	68.73	69.32
	225,692 (65%)	67.39	75.86	67.39	68.06
	173,609 (50%)	66.15	75.56	66.15	66.96

Table 4. *Cont.*

Feature Pattern	No. of Features	Accuracy	Precision	Recall	F1-Measure
T,Q	349,996 (100%)	44.26	81.19	44.26	46.9
	279,997 (80%)	42.77	81.14	42.77	45.26
	227,497 (65%)	41.56	81.06	41.56	43.83
	174,998 (50%)	40.51	80.9	40.51	42.7
U,B,T	372,179 (100%)	84.5	85.22	84.5	84.49
	297,743 (80%)	84.61	85.3	84.61	84.59
	241,916 (65%)	84.48	85.17	84.48	84.47
	186,090 (50%)	84.44	85.06	84.44	84.41
U,B,T,Q	540,661 (100%)	84.6	85.28	84.6	84.59
	432,529 (80%)	84.8	85.43	84.8	84.78
	351,430 (65%)	84.61	85.25	84.61	84.6
	270,331 (50%)	84.67	85.23	84.67	84.65

Similarly, Table 5 shows the results obtained by applying the kNN classification technique to the same feature patterns previously discussed; in the same way, we report the sensitivity analysis for each feature pattern. Table 6 reports the results obtained by applying the SVM classification technique, while Table 7 reports the results obtained by applying the Random-Forest classification technique.

Table 5. Experimental results for kNN classifier.

Feature Pattern	No. of Features	Accuracy	Precision	Recall	F1-Measure
U	24,961 (100%)	79.08	79.68	79.08	79.21
	19,969 (80%)	75.37	78.26	75.37	75.73
	16,225 (65%)	38.9	84.77	38.9	36.53
	12,481 (50%)	20.43	68.05	20.43	14.89
B	165,704 (100%)	17.64	56.32	17.64	11.67
	132,563 (80%)	17.6	54.71	17.6	11.33
	107,708 (65%)	16.93	45.3	16.93	13.31
	82,852 (50%)	19.83	51.72	19.83	14.65
T	181,514 (100%)	16.28	29.99	16.28	9.51
	145,211 (80%)	12.08	39.05	12.08	7.13
	117,984 (65%)	17.61	57.99	17.61	10.29
	90,757 (50%)	16.99	47.62	16.99	11.98
Q	168,482 (100%)	15.34	34.68	15.34	5.81
	134,786 (80%)	14.71	49.57	14.71	6.33
	109,513 (65%)	15.82	32.81	15.82	7.86
	84,241 (50%)	15.66	56.46	15.66	6.29
U,B	190,665 (100%)	78.95	79.25	78.95	79.03
	152,532 (80%)	78.51	79.32	78.51	78.71
	123,932 (65%)	76.34	78.76	76.34	76.81
	95,333 (50%)	75.72	78.49	75.72	76.16
B,T	347,218 (100%)	16.16	26.71	16.16	10.41
	277,774 (80%)	16.87	56.63	16.87	11.31
	225,692 (65%)	16.08	43.69	16.08	11.79
	173,609 (50%)	16	44.39	16	13.2
T,Q	349,996 (100%)	15.7	27.41	15.7	8.64
	279,997 (80%)	16.43	41.75	16.43	8.44
	227,497 (65%)	15.28	48.77	15.28	8.99
	174,998 (50%)	16.69	43.51	16.69	11.05

Table 5. *Cont.*

Feature Pattern	No. of Features	Accuracy	Precision	Recall	F1-Measure
U,B,T	372,179 (100%)	78.59	79.1	78.59	78.72
	297,743 (80%)	77.95	79.1	77.95	78.22
	241,916 (65%)	75.25	78.31	75.25	75.83
	186,090 (50%)	75.28	78.43	75.28	75.77
U,B,T,Q	540,661 (100%)	78.19	78.99	78.19	78.36
	432,529 (80%)	77.28	78.93	77.28	77.62
	351,430 (65%)	74.23	78.02	74.23	74.91
	270,331 (50%)	74.36	77.82	74.36	74.86

Table 6. Experimental results for Support Vector Machines (SVM) classifier.

Feature Pattern	No. of Features	Accuracy	Precision	Recall	F1-Measure
U	24,961 (100%)	85.29	85.67	85.29	85.36
	19,969 (80%)	85.46	85.85	85.46	85.52
	16,225 (65%)	85.33	85.73	85.33	85.4
	12,481 (50%)	85.36	85.73	85.36	85.42
B	165,704 (100%)	66.21	74.62	66.21	68.05
	132,563 (80%)	66.35	74.28	66.35	68,34
	107,708 (65%)	66.34	74.17	66.34	68.31
	82,852 (50%)	66.39	74.11	66.39	68.26
T	181,514 (100%)	42.14	73.48	42.14	45.66
	145,211 (80%)	42.08	73.23	42.08	45.47
	117,984 (65%)	42.73	73.62	42.73	45.96
	90,757 (50%)	43.94	76.51	43.94	47.19
Q	168,482 (100%)	30.39	72.93	30.39	30.95
	134,786 (80%)	30.53	81.22	30.53	30.57
	109,513 (65%)	30.26	81.8	30.26	30.25
	84,241 (50%)	29.99	82.4	29.99	30.08
U,B	190,665 (100%)	84.98	85.28	84.98	85.02
	152,532 (80%)	85.14	85.43	85.14	85.17
	123,932 (65%)	85.26	85.53	85.26	85.29
	95,333 (50%)	85.3	85.57	85.3	85.34
B,T	347,218 (100%)	64.39	74.26	64.39	66.53
	277,774 (80%)	64.82	74.37	64.82	67.18
	225,692 (65%)	64.9	74.13	64.9	67.18
	173,609 (50%)	65.62	73.93	65.62	67.66
T,Q	349,996 (100%)	39.3	73	39.3	42.65
	279,997 (80%)	39.68	73.01	39.68	43.04
	227,497 (65%)	41.09	73.27	41.09	44.31
	174,998 (50%)	43.36	79.93	43.36	46.24
U,B,T	372,179 (100%)	84.02	84.35	84.02	84.05
	297,743 (80%)	84.45	84.74	84.45	84.47
	241,916 (65%)	84.61	84.91	84.61	84.64
	186,090 (50%)	84.97	85.26	84.97	85.01
U,B,T,Q	540,661 (100%)	83.48	83.88	83.48	83.52
	432,529 (80%)	83.39	83.68	83.39	83.39
	351,430 (65%)	84.37	84.67	84.37	84.4
	270,331 (50%)	83.85	84.13	83.85	83.86

Table 7. Experimental results for Random-Forest classifier.

Feature Pattern	No. of Features	Accuracy	Precision	Recall	F1-Measure
U	24,961 (100%)	78.29	78.74	78.29	78.36
	19,969 (80%)	78.69	79.3	78.69	78.83
	16,225 (65%)	78.64	79.07	78.64	78.71
	12,481 (50%)	78.14	78.67	78.14	78.25
B	165,704 (100%)	60.37	73.73	60.37	62.75
	132,563 (80%)	60.42	75.13	60.42	63.17
	107,708 (65%)	60.66	74.7	60.66	63.33
	82,852 (50%)	60.69	74.67	60.69	63.35
T	181,514 (100%)	39.79	81.8	39.79	42.27
	145,211 (80%)	39.08	82.16	39.08	41.49
	117,984 (65%)	39.53	81.79	39.53	42.16
	90,757 (50%)	39.48	81.18	39.48	42.11
Q	168,482 (100%)	28.25	83.95	28.25	27.39
	134,786 (80%)	28.09	83.59	28.09	27.13
	109,513 (65%)	28.16	83.94	28.16	27.26
	84,241 (50%)	28.4	82.66	28.4	27.58
U,B	190,665 (100%)	77.03	77.88	77.03	77.17
	152,532 (80%)	77.79	78.49	77.79	77.92
	123,932 (65%)	77.95	78.79	77.95	78.11
	95,333 (50%)	78.3	79.1	78.3	78.43
B,T	347,218 (100%)	59.41	74.29	59.41	62.03
	277,774 (80%)	59.78	75.74	59.78	62.66
	225,692 (65%)	59.72	75.16	59.72	62.59
	173,609 (50%)	60.19	75.39	60.19	63.03
T,Q	349,996 (100%)	38.95	82.91	38.95	41.28
	279,997 (80%)	37.95	82.65	37.95	40.08
	227,497 (65%)	38.41	82.92	38.41	40.74
	174,998 (50%)	38.32	82.07	38.32	40.67
U,B,T	372,179 (100%)	76.12	77.28	76.12	76.3
	297,743 (80%)	76.6	77.63	76.6	76.78
	241,916 (65%)	77.58	79.03	77.58	77.82
	186,090 (50%)	77.23	78.34	77.23	77.42
U,B,T,Q	540,661 (100%)	76.03	77.36	76.03	76.18
	432,529 (80%)	75.45	77.64	75.45	75.61
	351,430 (65%)	76.87	78.29	76.87	77.09
	270,331 (50%)	76.94	78.53	76.94	77.06

Figure 4 depicts the results obtained by each classifier for all feature patterns by using the full set of features extracted from the training set. The blue line depicts the results obtained by the Naïve Bayes classifier; the red line depicts the results obtained by the kNN classifier; the brown line depicts the results obtained by the SVM classifier, the black line depicts the results obtained by the Random-Forest classifier.

We can notice that the Naïve Bayes classifier (blue line) always performed as the best classifier, always obtaining the highest accuracy. The SVM classifier (brown line) performed only a little bit worse, but results were comparable. The Random Forest classifier still showed comparable accuracy, even though a little bit less than Naïve Bayes and SVM classifiers.

In contrast, the inability of the kNN classifier to exploit most of feature patterns was evident. In details, we noticed that for **U**, **U,B**, **U,B,T** and **U,B,T,Q** feature patterns, the kNN classifiers obtained results that were comparable with the other classifiers. Instead, for feature patterns that did not include uni-grams, the kNN classifier obtained very poor results.

Nevertheless, notice that the other three tested classification techniques suffered for the absence of uni-grams in the feature sets as well, even though they behaved better than the kNN classifier.

If we focus on results obtained by each classifier for feature patterns that contain uni-grams, it clearly appears that no advantage was obtained by combining uni-grams with other features. Looking at Table 4, we see that the Naïve Bayes classifier obtained a very slight improvement; in contrast, looking at Tables 5–7, we can see a slight deterioration of accuracy, when comparing the **U** pattern with **U,B**, **U,B,T** and **U,B,T,Q** combined patterns.

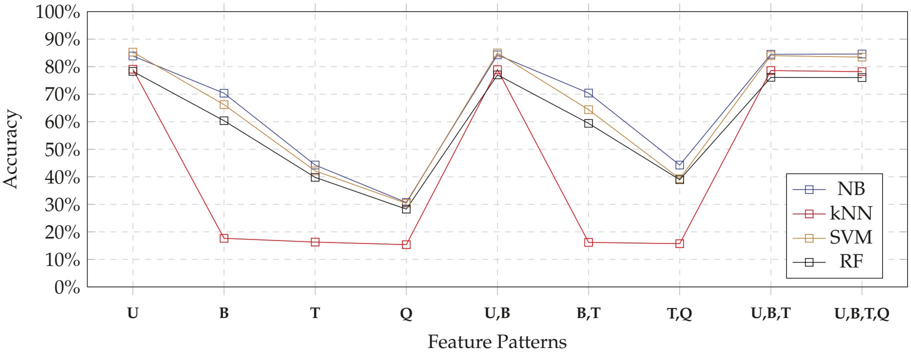


Figure 4. Comparing accuracy of classifiers for different feature patterns with 100% features.

5.3. Sensitivity Analysis

We can now consider the sensitivity analysis we performed. Recall that, apart from the full set of features, we also considered the best 80%, 65% and 50% of features, on the basis of the weight defined in Formula 1.

Figures 5–7 depict the results so far obtained, respectively, with the 80%, 65% and 50% of features. In the 80% case (Figure 5), no significant variations appeared: the performances obtained by all classifiers were, more or less the same. This is also confirmed by looking at the tables, that show very small reductions of accuracy. Nevertheless, the general behavior of the four classifiers remained exactly the same as for the full set of features. Consequently, we could argue that it was not the case to use the full set of features for training the classifiers, so as to save time and computational power.

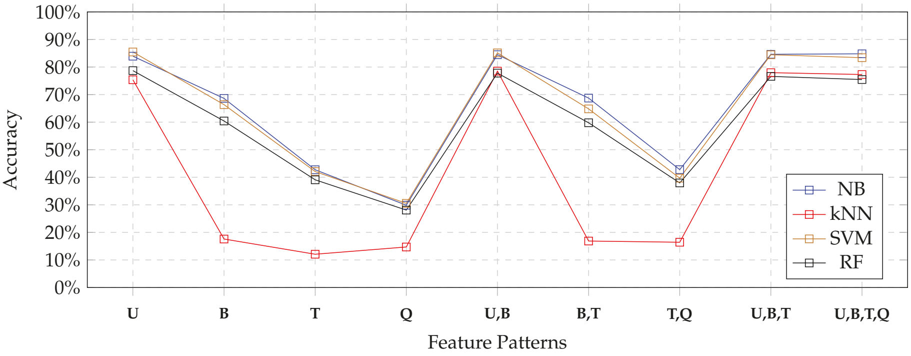


Figure 5. Comparing accuracy of classifiers for different feature patterns with 80% features.

Considering the 65% case (depicted in Figure 6), and the 50% case (depicted i Figure 7), we still observed a very slight reduction of accuracy. Only the kNN classifier behaved significantly worse with uni-gram patterns in the 50% case; nevertheless, looking at Figure 7, we notice that with patterns

U,B,T and **U,B,T,Q**, the combined feature patterns that contained uni-grams helped the classifier to obtain good results.

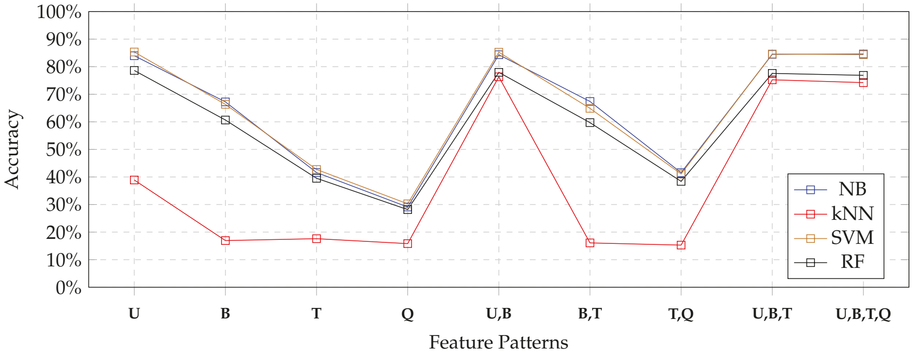


Figure 6. Comparing accuracy of classifiers for different feature patterns with 65% features.

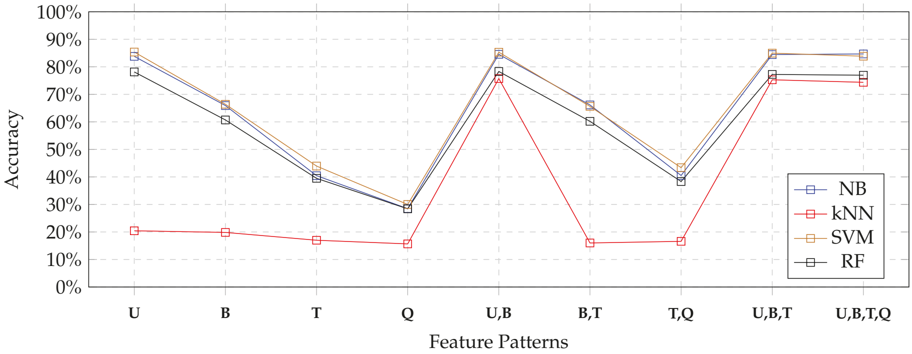


Figure 7. Comparing accuracy of classifiers for different feature patterns with 50% features.

In effect, looking at Table 5, we can see that both precision and recall strongly penalized the kNN classifier, with respect to the other competitors. This happened with all feature patterns.

5.4. Execution Times and Suitability Metric

Based on accuracy, the kNN classifier was not suitable for the investigated application context, while the other classifiers showed comparable performance. However, the cost of computation is an important issue, thus we also gathered execution times both for training and testing.

We performed experiments on a PC powered by Processor Intel(R) Core(TM) i7-5600U, with clock frequency of 2.60 GHz, equipped with 8 GB RAM; the operating system was Windows 10 Pro (64 bit).

Table 8 reports the execution times shown by the four classifiers on the full set of features, for the most promising feature patterns, i.e., **U**, **U,B**, **U,B,T** and **U,B,T,Q**. Specifically, we evaluated execution times during the training phase and during the test phase; notice that we also measured the execution times concerned with feature extraction, so as to understand how heavy the computation of Cartesian products of features was.

The first thing we can notice is that feature extraction was performed in a negligible time, if compared with the actual training performed by the classifier; even in the case of the most complicated feature pattern (i.e., **U,B,T,Q**), this time was negligible. Nonetheless, to obtain a given feature pattern, experiments confirmed that the library we adopted was deterministic, since the execution time was independent of the specific attempt.

Table 8. Execution times (in seconds) for the most promising feature patterns.

		NB		
Feature Pattern	**Feature Extraction Time**	**Model Building Time**	**Total Training Time**	**Testing Time**
U	0.0028	0.3626	0.3654	0.1572
U,B	0.0162	1.2932	1.3094	0.2905
U,B,T	0.0299	2.3124	2.3423	0.4124
U,B,T,Q	0.0434	3.2504	3.2938	0.4971
		kNN		
Feature Pattern	**Feature Extraction Time**	**Model Building Time**	**Total Training Time**	**Testing Time**
U	0.0028	0.3339	0.3367	5.1004
U,B	0.0162	1.2770	1.2932	5.0536
U,B,T	0.0299	2.2337	2.2636	5.7653
U,B,T,Q	0.0434	3.2577	3.3011	5.3663
		SVM		
Feature Pattern	**Feature Extraction Time**	**Model Building Time**	**Total Training Time**	**Testing Time**
U	0.0028	137.9433	137.9461	30.2422
U,B	0.0162	201.2421	201.2583	45.3217
U,B,T	0.0299	247.0215	247.0514	57.4804
U,B,T,Q	0.0434	281.5109	281.5543	62.1898
		RF		
Feature Pattern	**Feature Extraction Time**	**Model Building Time**	**Total Training Time**	**Testing Time**
U	0.0028	56.5771	56.5799	1.4356
U,B	0.0162	219.9926	220.0088	2.0941
U,B,T	0.0299	399.2106	399.2405	2.4720
U,B,T,Q	0.0434	617.9247	617.9681	2.6590

In contrast, looking at the execution time for model building, the reader can see that there were significant differences. Consequently, in order to choose the best classifier, the cost of computation should be considered. For this reason, we defined a cost-benefit metric, in such a way accuracy represents the benefit, while execution time represents the cost. We called this metric *Suitability*, because by means of it we wanted to rank classifiers in order to find out the one that was suitable for our context.

Consider a pool of experiments $E = \{e_1, e_2, \ldots, e_h\}$, where for an experiment e_i we refer to its accuracy as $e_i.Accuracy$, to its training execution times as $e_i.trtime$ (the execution time shown during the training phase) and to the test execution times as $e_i.tetime$ (the execution time shown during the test phase). The *Training Suitability* of an experiment is defined as

$$TrainingSuitability(e_i) = e_i.Accuracy \times \frac{mintrtime}{mintrtime + (e_i.trtime - mintrtime) \times \beta} \quad (3)$$

where $mintrtime = min_{\forall e_i \in E}(e_i.trtime)$ (i.e., the minimum training execution time). β is a importance weight of the difference between the training execution time of the e_i experiment and the minimum training execution time; we decided to set it to 50%, in order to mitigate the effect of execution times on the final score; in fact, with $\beta = 1$, the penalty effect would be excessive.

Similarly, we can define the *Testing Suitability*, defined in Formula 4.

$$TestingSuitability(e_i) = e_i.Accuracy \times \frac{mintetime}{mintetime + (e_i.tetime - mintetime) \times \gamma} \quad (4)$$

where $mintetime = min_{\forall e_i \in E}(e_i.tetime)$ (i.e., the minimum testing execution time among the experiments). Similarly to β, γ is the relevance of the difference between execution time of the e_i experiment and the minimum testing time. We decided to set it to 50% as well.

Training Suitability and *Testing Suitability* ranked experiments by keeping the two phases (training and testing) separated.

In Formula 5, we propose a unified *Suitability* metric.

$$Suitability(e_i) = \alpha \times TrainingSuitability(e_i) + (1 - \alpha) \times TestingSuitability(e_i) \tag{5}$$

i.e., the unified suitability is the weighted average of *Training Suitability* and *Testing Suitability*, where $\alpha \in [0, 1]$ balances the two contributions.

Table 9 reports the values of training suitability, testing suitability and unified suitability for the same experiments considered in Table 8. Figure 8 depicts the results, by using the same convention as in Figure 4, by using the unified suitability. The reader can see that the Naive Bayes classifier had the highest suitability values, due to its ability to combine high accuracy and very low execution times. Surprisingly, the kNN classifier obtained the second position; in fact, in spite of the fact that it obtained the worst accuracy values, it obtained the lowest execution times. Finally, the SVM classifier and the Random-Forest classifier were strongly penalized by their execution times.

Table 9. Suitability for the most promising feature patterns.

Feature Pattern	Classifier	*TrainingSuitability*	*TestingSuitability*	*Suitability*
U	**NB**	80.4512	83.8872	82.1692
U,B		34.4861	59.1987	46.8424
U,B,T		21.2401	46.6448	33.9425
U,B,T,Q		15.6920	40.6500	28.1710
U	**kNN**	79.0800	4.7289	41.9044
U,B		32.6185	4.7653	18.6910
U,B,T		20.3525	4.1720	12.2622
U,B,T,Q		14.4739	4.4506	9.4623
U	**SVM**	0.4153	0.8821	0.6487
U,B		0.2839	0.5875	0.4357
U,B,T		0.2287	0.4583	0.3435
U,B,T,Q		0.1994	0.4210	0.3102
U	**RF**	0.9263	15.4535	8.1899
U,B		0.2354	10.7576	5.4965
U,B,T		0.1283	9.1025	4.6154
U,B,T,Q		0.0828	8.4880	4.2854

Consequently, on the basis of Table 9 and Figure 8, we can clearly say that the Naive Bayes classifier applied to the feature pattern of uni-grams clearly emerged as the most suitable solution for our application context (presented in Section 3) and specifically to solve Problem 1.

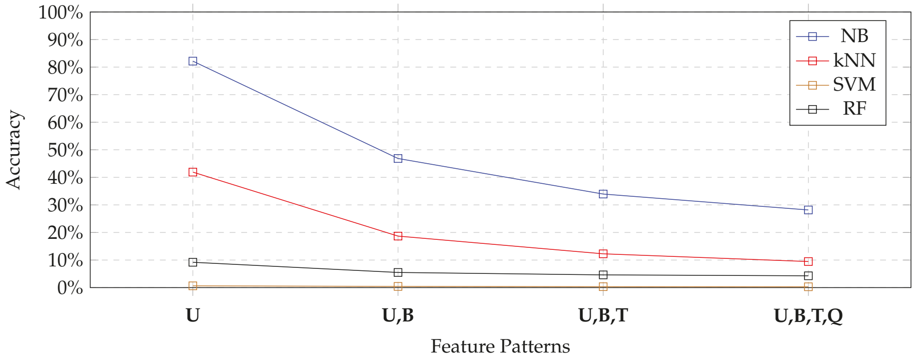


Figure 8. Suitability for the most promising feature patterns.

6. Conclusions and Future Work

In this paper, we have posed the basic brick towards the extension of micro-blog user interfaces with a new functionality: a tool to recommend users with other users to follow (influencers) on the basis of topics their message talk about. The basic brick is a text classification technique applied to a given feature pattern that provides good accuracy by requiring limited execution times. To identify it, we built an investigation framework, that allowed us to perform experiments, by measuring effectiveness (accuracy) and execution times. A cost-benefit function, called *Suitability*, has been defined: by means of it, we discovered that the best solution to address the problem is to apply a Naive Bayes classifier to uni-grams extracted from within messages, both to train the model and to classify unlabeled messages. We considered execution times because, in our opinion, the envisioned application scenario asks for fast functionalities; thus execution times emerge as critical factors. At the best of our knowledge, this comparative study of performances shown by classifiers, based on both accuracy and execution times, is a unique contribution of this paper.

The next steps towards the more ambitions goal of building a recommender system for influencers is to develop the surrounding methodology that actually enables to recommend influencers: in fact, once messages are labeled with topics, it is necessary to rank potential influencers, on the basis of the frequency with which they post messages about a given topic. This methodology will be the next step of our work.

Author Contributions: Conceptualization and methodology, G.P.; software, M.A., A.B.; writing—original draft preparation, M.A.; writing—review and editing, G.P.; Data collection and annotation, S.M., A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Twitter. About(Twitter). 2019. Available online: https://about.twitter.com/company (accessed on 26 July 2019)
2. Sharma, R.; Uniyal, S.; Gera, V. Performing Interest Mining on Tweets of Twitter Users for Recommending Other Users with Similar Interests. In *Progress in Advanced Computing and Intelligent Engineering*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 593–603.
3. Kiruthika, M.; Woonna, S.; Giri, P. Sentiment analysis of twitter data. *Int. J. Innov. Eng. Technol.* **2016**, *6*, 264–273.
4. Zhao, Y. Twitter Data Analysis with R-Text Mining and Social Network Analysis. In *Short Course on R and Data Mining*; University of Canberra: Canberra, Australia, 2016.
5. Cuzzocrea, A.; Psaila, G.; Toccu, M. An innovative framework for effectively and efficiently supporting big data analytics over geo-located mobile social media. In Proceedings of the 20th International Database Engineering & Applications Symposium, Montreal, QC, Canada, 12–14 July 2016; pp. 62–69.
6. Bordogna, G.; Frigerio, L.; Cuzzocrea, A.; Psaila, G. An effective and efficient similarity-matrix-based algorithm for clustering big mobile social data. In Proceedings of the 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016; pp. 514–521.
7. Bordogna, G.; Cuzzocrea, A.; Frigerio, L.; Psaila, G.; Toccu, M. An interoperable open data framework for discovering popular tours based on geo-tagged tweets. *Int. J. Intell. Inf. Database Syst.* **2017**, *10*, 246–268. [CrossRef]
8. Jain, S.; Sharma, V.; Kaushal, R. PoliticAlly: Finding political friends on twitter. In Proceedings of the 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), Kolkata, India, 15–18 December 2015; pp. 1–3.
9. Deitrick, W.; Valyou, B.; Jones, W.; Timian, J.; Hu, W. Enhancing sentiment analysis on twitter using community detection. *Commun. Netw.* **2013**, *5*, 192.

10. Mirani, T.B.; Sasi, S. Sentiment analysis of ISIS related Tweets using Absolute location. In Proceedings of the 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15–17 December 2016; pp. 1140–1145.
11. Kanavos, A.; Nodarakis, N.; Sioutas, S.; Tsakalidis, A.; Tsolis, D.; Tzimas, G. Large scale implementations for twitter sentiment classification. *Algorithms* **2017**, *10*, 33. [CrossRef]
12. Hassan, S.U.; Aljohani, N.R.; Idrees, N.; Sarwar, R.; Nawaz, R.; Martínez-Cámara, E.; Ventura, S.; Herrera, F. Predicting literature's early impact with sentiment analysis in Twitter. *Knowl. Based Syst.* **2019**, *192*, 105383.[CrossRef]
13. Halibas, A.S.; Shaffi, A.S.; Mohamed, M.A.K.V. Application of text classification and clustering of Twitter data for business analytics. In Proceedings of the 2018 Majan International Conference (MIC), Muscat, Oman, 19–20 March 2018; pp. 1–7.
14. Chang, Y. Spectators' emotional responses in tweets during the Super Bowl 50 game. *Sport Manag. Rev.* **2019**, *22*, 348–362.[CrossRef]
15. D'Andrea, E.; Ducange, P.; Bechini, A.; Renda, A.; Marcelloni, F. Monitoring the public opinion about the vaccination topic from tweets analysis. *Expert Syst. Appl.* **2019**, *116*, 209–226. [CrossRef]
16. Geetha, S.; Kumar, K.V. Tweet Analysis Based on Distinct Opinion of Social Media Users'. In *Advances in Big Data and Cloud Computing*; Springer: Singapore, 2019; pp. 251–261.
17. Razzaq, M.A.; Qamar, A.M.; Bilal, H.S.M. Prediction and analysis of Pakistan election 2013 based on sentiment analysis. In Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), Beijing, China, 17–20 August 2014; pp. 700–703.
18. Dubey, G.; Chawla, S.; Kaur, K. Social media opinion analysis for indian political diplomats. In Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, Noida, India, 12–13 January 2017; pp. 681–686.
19. Liu, B.; Hu, M.; Cheng, J. Opinion observer: analyzing and comparing opinions on the web. In Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, 10–14 May 2005; pp. 342–351.
20. Hasan, A.; Moin, S.; Karim, A.; Shamshirband, S. Machine learning-based sentiment analysis for twitter accounts. *Math. Comput. Appl.* **2018**, *23*, 11.
21. Liew, S.W.; Sani, N.F.M.; Abdullah, M.T.; Yaakob, R.; Sharum, M.Y. An effective security alert mechanism for real-time phishing tweet detection on Twitter. *Comput. Secur.* **2019**, *83*, 201–207. [CrossRef]
22. Washha, M.; Qaroush, A.; Mezghani, M.; Sedes, F. Unsupervised Collective-based Framework for Dynamic Retraining of Supervised Real-Time Spam Tweets Detection Model. *Expert Syst. Appl.* **2019**, *135*, 129–152.[CrossRef]
23. Bhargava, N.; Sharma, G.; Bhargava, R.; Mathuria, M. Decision tree analysis on j48 algorithm for data mining. *Proc. Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2013**, *3*, 1114–1119.
24. Ghaly, R.S.; Elabd, E.; Mostafa, M.A. Tweets classification, hashtags suggestion and tweets linking in social semantic web. In Proceedings of the 2016 SAI Computing Conference (SAI), London, UK, 13–15 July 2016; pp. 1140–1146.
25. Fiallos, A.; Jimenes, K. Using Reddit Data for Multi-Label Text Classification of Twitter Users Interests. In Proceedings of the 2019 Sixth International Conference on eDemocracy & eGovernment (ICEDEG), Quito, Ecuador, 24–26 April 2019; pp. 324–327.
26. Azam, N.; Yao, J. Comparison of term frequency and document frequency based feature selection metrics in text categorization. *Expert Syst. Appl.* **2012**, *39*, 4760–4768. [CrossRef]
27. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
28. Indra, S.; Wikarsa, L.; Turang, R. Using logistic regression method to classify tweets into the selected topics. In Proceedings of the 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Malang, Indonesia, 15–16 October 2016; pp. 385–390.
29. Jeong, O.R. SNS-based recommendation mechanisms for social media. *Multimed. Tools Appl.* **2015**, *74*, 2433–2447. [CrossRef]
30. Li, W.; Ye, Z.; Xin, M.; Jin, Q. Social recommendation based on trust and influence in SNS environments. *Multimed. Tools Appl.* **2017**, *76*, 11585–11602. [CrossRef]
31. Milgram, S. The small world problem. *Psychol. Today* **1967**, *2*, 60–67.
32. Chen, J.; Wang, C.; Shi, Q.; Feng, Y.; Chen, C. Social recommendation based on users' attention and preference. *Neurocomputing* **2019**, *341*, 1–9. [CrossRef]

33. Lai, C.H.; Lee, S.J.; Huang, H.L. A social recommendation method based on the integration of social relationship and product popularity. *Int. J. Hum. Comput. Stud.* **2019**, *121*, 42–57. [CrossRef]
34. Li, Y.; Liu, J.; Ren, J. Social recommendation model based on user interaction in complex social networks. *PLoS ONE* **2019**, *14*, e0218957. [CrossRef]
35. Li, W.; Ni, Y.; Wu, M.; Ye, Z.; Jin, Q. Social recommendation algorithm dynamically adaptable to user profiling for SNS. In Proceedings of the 2014 Second International Conference on Advanced Cloud and Big Data, Huangshan, China, 20–22 November 2014; pp. 261–266.
36. Tripathy, A.; Agrawal, A.; Rath, S.K. Classification of sentiment reviews using n-gram machine learning approach. *Expert Syst. Appl.* **2016**, *57*, 117–126. [CrossRef]
37. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [CrossRef]
38. Kadhim, A.I. Term Weighting for Feature Extraction on Twitter: A Comparison Between BM25 and TF-IDF. In Proceedings of the 2019 International Conference on Advanced Science and Engineering (ICOASE), Duhok, Iraq, 2–4 April 2019; pp. 124–128.
39. Wu, H.C.; Luk, R.W.P.; Wong, K.F.; Kwok, K.L. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.* **2008**, *26*, 1–37. [CrossRef]
40. Kantardzic, M. *Data Mining: Concepts, Models, Methods, and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
41. Alsaleem, S.; others. Automated Arabic Text Categorization Using SVM and NB. *Int. Arab J. Technol.* **2011**, *2*, 124–128.
42. Lee, Y.; Lin, Y.; Wahba, G. Multicategory Support Vector Machines, Theory, and Application to the Classification of Microarray Data and Satellite Radiance Data. *J. Atmos. Ocean. Technol.* **2003**, *99*, 67–81.
43. Al-Shalabi, R.; Obeidat, R. Improving kNN Arabic text classification with n-grams based document indexing. In Proceedings of the Sixth International Conference on Informatics and Systems, Cairo, Egypt, 27–29 March 2008; pp. 108–112.
44. Ho, T.K. Random Decision Forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995; pp. 278–282.
45. Psaila, G.; Toccu, M. A Fuzzy Technique for On-Line Aggregation of POIs from Social Media: Definition and Comparison with Off-Line Random-Forest Classifiers. *Information* **2019**, *10*, 388. [CrossRef]
46. Python. NLTK. 2019. Available online: https://www.nltk.org/ (accessed on 1 September 2019).
47. Porter, M.F.; others. An algorithm for suffix stripping. *Program* **1980**, *14*, 130–137. [CrossRef]
48. Twitter. Twitter Apps. 2019. Available online: http://www.tweepy.org (accessed on 25 August 2019).
49. Wang, Q.; Bhandal, J.; Huang, S.; Luo, B. Content-based classification of sensitive tweets. *Int. J. Semant. Comput.* **2017**, *11*, 541–562. [CrossRef]
50. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
51. Zhang, M.L.; Zhou, Z.H. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.* **2013**, *26*, 1819–1837. [CrossRef]

applied sciences

Article

Time-Aware Learning Framework for Over-The-Top Consumer Classification Based on Machine- and Deep-Learning Capabilities

Jaeun Choi [1] and Yongsung Kim [2,*]

[1] Department of Artificial Intelligence Software, Kyungil University, Gyungbuk 38428, Korea; juchoi@kiu.kr
[2] Department of Software Engineering, The Cyber University of Korea, Seoul 03051, Korea
* Correspondence: kys1001@cuk.edu; Tel.: +82-2-6361-1948

Received: 29 October 2020; Accepted: 26 November 2020; Published: 27 November 2020

Abstract: With the widespread use of over-the-top (OTT) media, such as YouTube and Netflix, network markets are changing and innovating rapidly, making it essential for network providers to quickly and efficiently analyze OTT traffic with respect to pricing plans and infrastructure investments. This study proposes a time-aware deep-learning method of analyzing OTT traffic to classify users for this purpose. With traditional deep learning, classification accuracy can be improved over conventional methods, but it takes a considerable amount of time. Therefore, we propose a novel framework to better exploit accuracy, which is the strength of deep learning, while dramatically reducing classification time. This framework uses a two-step classification process. Because only ambiguous data need to be subjected to deep-learning classification, vast numbers of unambiguous data can be filtered out. This reduces the workload and ensures higher accuracy. The resultant method provides a simple method for customizing pricing plans and load balancing by classifying OTT users more accurately.

Keywords: consumer classification; deep learning; machine learning; over-the-top; time-aware classification

1. Introduction

With the advancements of smart devices and the rapid development of wired and wireless networks, our modes of entertainment and venues for information are changing rapidly. In the past, our receipt of multimedia mainly relied upon standardized broadcasts from franchise television (TV) networks. With the turn of the century, our multimedia consumption has centered around internet smartphones, as typified by the "over-the-top" internet-protocol (IP) services of YouTube and Netflix [1]. Because of its ubiquity and simplicity, OTT content can be viewed worldwide without TVs. According to PricewaterhouseCoopers, the global OTT market is expected to grow sharply from USD 81.6 B in 2019 to USD 156.9 B in 2024 with an annual average growth of approximately 14% [2]. Furthermore, subscription OTT services are expected to reach approximately 650 million by 2021 [3]. With this growth, the dominant OTT players such as YouTube, Netflix, Amazon Prime, and Hulu, are actively promoting entry into the global market, and Disney, Apple, Warner Media, and HBO are gaining access using their existing content. The rising OTT market is indeed becoming a fierce battleground.

Consequently, network broadcast media and more recent TV-over-IP enterprises have experienced heavy competition [4,5]. On the other hand, internet-service providers (ISP) increasingly find themselves in conflicts with OTT media providers because of bandwidth- and fee-related issues. Because OTT services consume a vast amount of network resources, ISPs seek to bolster their profits to support growth [6]. Furthermore, consumers increasingly demand higher quality of service (QoS) from their ISPs [7] who desperately need to expand their throughput capabilities [8]. Often, the ISPs respond to bandwidth overloads by limiting the amount of throughput (i.e., throttling) based on the OTT service in demand. This process is most often reactionary, inevitably creating surges of consumer complaints. To balance this vicious cycle of competing demands, the ISPs require better and more-timely consumer OTT-usage analysis capabilities, so that they can better mitigate network performance issues while balancing customer demand. Furthermore, utilizing a sound and trustworthy tool such for this purpose would put the ISPs in a better position to negotiate with OTT providers [9].

Traffic analysis models have been widely researched and utilized for this purpose in conventional hypertext-transfer-protocol (HTTP) mobile-network environments. Machine learning has been key to their success. However, only a few academic studies have focused on OTT content in the context of strategic service provision [9]. In this study, we analyze network consumption patterns based on consumers' OTT-usage patterns, confirming that a combination of machine- and deep-learning capabilities can achieve the highest accuracy. Additionally, by mitigating the time and resource requirements of deep learning, we provide a novel MetaCost-based framework related to OTT user analysis that can reduce the time required for analysis while exploiting the technology's high accuracy. This framework drastically reduces the analysis workload, making the process very efficient and timely so that ISPs can achieve instantaneous status and influence over OTT service demands.

This paper is structured as follows. In Section 2, we examine why the analysis of OTT-related trends and data-usage patterns is critical. We also justify the application of machine and deep learning to this pursuit with a review of previous traffic-analysis studies. In Section 3, we fully describe the OTT user-analysis framework. Section 4 presents a discussion of our experimental results. Finally, Section 5 presents the conclusions of this study with further research directions.

2. Literature Review

2.1. OTT Services

The success of OTT services is owed, in part, to the increase in the number of single-person households and their desire for highly personalized content. With the phenomena of cord-cutting, which circumvents paid broadcasting, and cord-shaving, which leverages alternate broadcasting venues, the demand for new and innovative OTT services will not likely decrease [10]. For many worldwide consumers, OTT services have replaced legacy subscription models. In Korea and China, consumers spend around USD 3 per month for high-definition OTT services with tailored recommendation systems [11].

Currently, the OTT market is dominated by giant companies such as Netflix, YouTube, Amazon Prime Video, and Hulu, which account for approximately 75, 55, 44, and 32% of the US OTT market, respectively. As a whole, these firms currently account for 79% of the US market share [12]. Competitors are quickly entering the fray. Disney launched Disney+ in November 2019 after acquiring 21st Century Fox, securing 50-million US subscribers as of July 2020 [13]. After acquiring Warner Media, AT&T launched HBO Max, which utilizes current and past HBO content, securing more than 34-million US subscribers as of June 2020 [14].

The rapidly changing OTT landscape presents both a crisis and an opportunity for conventional TV networks. Although it proves to be a disadvantage to extant strategies, it does provide an opportunity to enter new markets by providing OTT services using their current content and service supply chains [5]. Hulu, launched in 2008, dominates the market with content from FOX, NBC, and ABC [4]. In Korea, there are ~3-million monthly Wavve subscribers, which offers content from KBS, MBC, and SBS [15].

The market positions and strategies of ISPs are more complicated than those of legacy TV networks. ISPs that simultaneously provide IP-TV and internet services must install and maintain high-quality broadband infrastructures for both. Additionally, bundling strategies are required to prevent cord-cutting that would cancel IP-TV services in favor of OTT services alone [16]. In fact, in Korea, KT, which holds the highest share of the IP-TV market, is considering a strategy to create synergy through a partnership with Netflix. As such, Netflix could use the opportunity to expand their market further into Korea. KT has the largest number of wired network subscribers and can collect more subscribers and their abundant network usage fees by providing Netflix content [17]. With the gradual distribution of 5G, the number of customers using wireless OTT services is bound to grow. If stable QoS cannot be guaranteed, customer churn will be difficult to deal with. In particular, with the increase of the use of real-time video-streaming services (e.g., Twitch and Discord) stable services will forever be challenged [8,18].

As mentioned, ISPs must be able to dynamically execute service degradation plans to minimize network-resource overconsumption while meeting the QoS needs of consumers. That is, it is essential to create a win–win situation for both ISPs and OTT providers. Based on assumptions of network neutrality, various studies have analyzed the complex pricing systems related to content providers, networks, and consumers. Dai et al. [7] proposed a pricing plan that could guarantee QoS based on the Nash equilibrium. They showed that the direct sale of QoS by an ISP to a consumer achieved better results than selling QoS to an OTT provider. Based on the quality of experience (QoE), a model that would benefit all OTT providers, networks, and consumers was also proposed [19]. This study compared three methods: Providing better QoE to customers that paid more; satisfying QoEs of the most profitable customers (MPC) to increase lifetime value; and providing fair QoE to all customers. Of these, the method of providing QoE to MPCs was found to be the most beneficial. A study based on shadow pricing was also conducted to determine an effective method to price broadband services [20], concluding that setting the pricing plan according to the usage patterns of consumers was the best strategy. Because OTT services utilize a considerable amount of network bandwidth, some studies have proposed methods of predicting network consumption and pricing via a content-delivery or a software-defined network [21,22]. With the spread of OTT services, the pricing-related issues of OTT and network providers persist, and most existing studies have suggested plans based on the amount of network usage. Thus, in order for networks or OTT providers to establish an optimal strategy related to OTT, the OTT-service usage patterns of consumers must be identified very quickly. Both network and OTT providers can establish effective pricing strategies only when they can correctly and immediately identify which users are using what OTT services and how much data they consume, classifying all items and load-balancing accordingly. Thus, OTT user classification is the first step in establishing an effective strategy.

2.2. *Review of Classification Using Machine Learning*

Researchers have conducted extensive studies on methods to manage and operate networks by analyzing network traffic and user behaviors. Hence, the widespread use of network technologies incorporating artificial-intelligence (AI) technologies has gained attention. Extensive research has been conducted on the knowledge-defined-networking paradigm, in which AI technology is incorporated into network routing, resource management, log analysis, and planning. Several companies have already applied AI data analysis to network operations [23]. In turn, multiple studies have been conducted to determine how network providers can leverage machine learning to analyze user traffic. Middleton and Modafferi [24] exploited machine learning to classify IP traffic in support of QoS guarantees. Yang et al. [25] proposed the classification of Chinese mobile internet users into heavy and high-mobility users by analyzing the network traffic of 2G and 3G services. Various other studies proposed methods, such as decision trees [26,27], support-vector machine (SVM) [28–30], *k*-nearest neighbor (KNN) [31,32], hidden Markov model (HMM) [33,34], and K-Means [35,36] for traffic classification. The application targets of these methods differed depending on whether the analysis was performed on wired, wireless, or encrypted traffic. However, these techniques demonstrated the following structure: They captured and analyzed network traffic; they applied machine learning by using traffic characteristics as features; and they classified the traffic data. The traffic data were diverse, ranging from captured packets to public datasets. However, data related to OTT usage, which is a recent trend, were rarely considered. Few studies have proposed methods to classify users based on OTT consumption [9,37]. Those that did classified consumers into three consumption categories (i.e., high, medium, and low) by using various machine-learning methods to analyze OTT traffic. The current study is significant in that it is the first to use deep learning in an attempt to classify users in terms of OTT usage. However, deep learning has the disadvantage of requiring large numbers of calculations. On the other hand, it has the advantage of high accuracy. Hence, it has been widely utilized for similar classification problems [38–40]. Therefore, in this study, to overcome the demerits of excessive time-consumption, we propose a time-aware user-analysis framework that applies the MetaCost method [41]. Based on Bayes risk theory, MetaCost can reduce specific classification errors while setting the cost of misclassification differently. By using these properties, we can reduce the load on deep learning through cost adjustment.

3. Research Design

First, we verified whether OTT users can be effectively classified using machine- and deep-learning methods. The description and application method of the machine- and deep- learning used in this study are detailed in Section 3.1. In addition, we were able to confirm that the classification accuracy was high when using deep learning; however, it was also found that the time required was large due to the characteristics of deep learning. Therefore, in Section 3.2, we propose a framework that can reduce time consumption but utilizes the accuracy of deep learning as well.

3.1. Appling Machine- and Deep- Learning to OTT Consumer Classification

Figure 1 shows the process of analyzing OTT users based on machine- and deep- learning [42,43]. The steps include raw data collection, data preprocessing for feature extraction, and dataset processing for machine learning. We leverage OTT usage data for this purpose. Our dataset was previously published [37] and is open to the public. It includes general network traffic characteristics but is also appropriate for OTT-specific research. It contains traffic information about the activities of actual internet users with respect to 29 types of OTT services. A detailed description of the dataset is presented in Section 3.3.

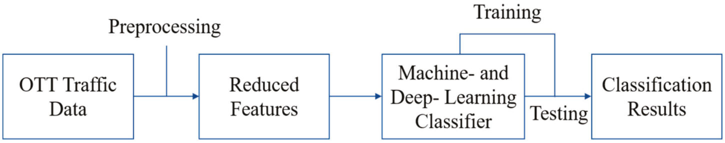


Figure 1. Machine- and deep-learning-based over-the-top (OTT) consumer classification process.

In this study, we analyze OTT users according to the conventional machine-learning methods of KNN, decision tree, SVM, naïve Bayes, and repeated incremental pruning to produce error reduction (RIPPER) models. We also use the multilayer perceptron (MLP) and convolutional neural network (CNN) as deep-learning applications.

3.1.1. Conventional Machine Learning Methods

The KNN is a typical classification method that applies clustering. It first confirms which class the k neighbors of a data point belong to, and it then performs classification by taking the majority vote based on the result. If $k = 1$, it is assigned to the nearest-neighbor class. Therefore, if k represents an odd number, classification becomes easier, because there is no possibility of a tie [44]. A neighbor in KNN can be defined by calculating the distance between vectors in a multidimensional feature set. The distance between vectors is calculated using Euclidean and Manhattan distances. Suppose that input-sample x has m features. The feature set of x is expressed as $(x_1, x_2, \cdots, x_m)$, and the Euclidean and Manhattan distances of x and y are defined as follows [45]:

$$\text{Euclidean}: \ D(x, y) = \sqrt{\sum\nolimits_{i=1}^{m} |x_i - y_i|^2}, \tag{1}$$

$$\text{Manhattan}: \ D(x,y) = \sum\nolimits_{i=1}^{m} \left|x_i - y_i\right|. \tag{2}$$

The greatest advantage of the KNN is its straightforwardness, and the variables are not required to be adjusted. This method only requires the assignment of a k value. However, if the distribution is distorted, KNN cannot be applied, because the data may not belong to the same class, even if it is close to its neighbors. Additionally, when dealing with multidimensional data having many features, classification accuracy may be degraded, owing to the curse of dimensionality. Thus, it is essential to reduce features [46].

Decision trees are built upon the tree model and re-used to classify an entire dataset into several subgroups. When traversing from an upper to a lower node, nodes are split according to the classification variables. Furthermore, nodes of the same branch have similar attributes, whereas nodes of different branches have different attributes. The most typical decision-tree algorithms are ID3 and C4.5. The C4.5 algorithm, derived from ID3, minimizes the entropy sum of the subsets by leveraging the concept of "information gain". The subset is split to the direction that maximizes information gain. Thus, accuracy is high when classification is performed through the learned result. Decision trees have the advantages of intuitiveness, high classification accuracy, and simple implementation. Therefore, they are widely adopted for various classification tasks. However, for data containing variables at different levels, the level is biased mainly to most of the data. Moreover, for a small tree with fewer branches, rule extraction is easy and intuitive, but accuracy may decrease. Moreover, for a deep and wide tree having many branches, rule extraction is difficult and non-intuitive, but accuracy may be higher than that of a small tree.

The SVM performs classification based on a hyperplane that separates two classes in the feature space. The hyperplane having the longest distance between the closest data points to the hyperplane in two classes is set to have the maximal margin. For inputs x and y, the hyperplane separating the classes is defined as $w^T \cdot x + b = 0$. After finding the distance between the hyperplane and closest data points of two classes, the optimization equation for maximization is defined as follows:

$$\min_{\mathrm{w},\, b} \Phi(\mathrm{w}) = \frac{1}{2} \|w\|^2, \tag{3}$$

where variables w and b, which satisfy the following convex quadratic programming, become variables that build the optimal hyperplane [45]:

$$\text{s.t } y_i\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}_i + b\right) \geq 1, \ i = 1, \cdots, l. \tag{4}$$

The SVM achieves high performance for a variety of problems. It is also known to be effective for the case of many features. Although SVM solves binary-class problems and can be applied to multiclass problems having various classes, it must solve multiple binary-class problems to derive accurate results. Thus, its calculation time is relatively long [45,46].

The naïve Bayes method is a typical classification method based on the statistical assumptions of the Bayes' theorem. It starts from the assumption that all input features are independent. When this is true, classes can be assigned through the following process [46]:

$$y(f_1, f_2, \cdots, f_m) = argmax_{k \in \{1,\cdots,K\}} p(C_k) \prod\nolimits_{i=1}^{m} p\left(f_i \middle| C_k\right), \tag{5}$$

where m is the number of features, k is the number of classes, f_i is the ith feature, C_k is the kth class, $p(C_k)$ is the prior probability for C_k, and $p\left(f_i \middle| C_k\right)$ is the conditional probability for feature f_i given class C_k.

The key merit of naïve Bayes is its short calculation time for learning. This is because, under the assumption that features are independent, high-dimensional density estimation is reduced to 1D kernel-density estimation. However, because the assumption that all features are independent is unrealistic, accuracy may decrease when performing classification using only a small sample of data. To increase accuracy, a large amount of data should be collected [46,47].

The RIPPER algorithm is a typical rule-set classification method [48]. Rules are derived by training the data using a separate-and-conquer algorithm. In turn, the rules are set up to cover as many datasets as possible, as developed using the current training data. The rules are pruned to maximize performance. Data correctly classified according to the rules are then removed from the training dataset [46]. The RIPPER algorithm overcomes the shortcoming of early rule algorithms, wherein big data could not be effectively processed. However, because the RIPPER algorithm starts by classifying two classes, performance can decrease when the number of classes increases. Its performance may also decrease because of its heuristic approach.

3.1.2. Deep Learning

After the AlphaGo (AI) beat Lee Se-dol (human) in the 2016 Google DeepMind Go challenge match, deep learning captured the attention of the worldwide public. However, research on deep learning had already been actively underway in academia and practical application fields. Deep learning is an extension of the artificial neural network, and it learns and makes decisions by configuring the number of layers that make up its neural network. It took a while for computer hardware to catch up, but with recent graphical processing-unit developments, deep learning has been widely applied in various fields. Deep learning automatically selects features through its training process. It does not require much assistance from domain experts and learns complex patterns of constantly evolving data [43]. As such, many related studies on internet traffic analysis have been published [49]. MLP and CNN deep-learning methods are used for this paper.

The MLP has a simple deep-learning structure and comprises an input layer, an output layer, and a hidden layer of neurons. Figure 2 shows the structure of the MLP. In each layer, several neurons are connected to the adjacent layer. The neurons calculate the weighted sum of inputs and output the results via a nonlinear activation function. In this process, the MLP uses a supervised back-propagation learning method. Because all nodes are connected within the MLP, each node in each layer has a specific weight, w_{ij}, with all nodes of the adjacent layer. Node weights are adjusted based on back-propagation, which minimizes the error of the overall result [49]. However, the MLP method has the disadvantage of being very complex and inefficient, owing to the huge number of variables the model must learn [43]. Accordingly, to use an MLP, it is necessary to acquire data that is not too complex or to pay close attention to time consumption. The OTT dataset used in this study has quantitative values for each feature. Since it does not have complicated feature structures such as images or videos, it shows sufficiently good performance even if only simple MLP is applied. Therefore, we tried to save the time required for learning and detection by using the simplest MLP structure possible. In this study, we conduct an experiment with an input layer, an output layer, and a single hidden layer between them.

The CNN is similar to the MLP, comprises several layers, and updates variables through learning. Although the MLP does not handle multi-dimensional inputs well, the CNN does so by applying a convolution layer. Figure 3 shows the structure of the CNN. The convolution layer produces results for the next layer by using kernels with learnable variables as inputs. The local filter is used to complete the mapping process, which is regarded as a convolution function. Additionally, because it is replicated in units, it shares the same weight vector and bias, thus increasing efficiency by greatly reducing the number of parameters. CNNs use a pooling process for down-sampling and can be widely applied to a variety of classifications. If the dimension of a vector used in the CNN process is 1, 2, or 3, it corresponds to 1D-, 2D-, and 3D-CNNs, respectively. The 1D-CNN is suitable for sequential data (e.g., language), the 2D-CNN is suitable for images or audio, and the 3D-CNN is suitable for video or large-volume images. Although there has been no research on classifying OTT traffic data by CNN, a study analyzing general network traffic via CNNs utilized a 1D-CNN [50]. This is because traffic characteristics are sequential; therefore, 1D-CNNs are sufficient and multi-dimensional CNNS are not required. In this study as well, OTT traffic was analyzed using a 1D-CNN since OTT traffic is similar to traditional traffic data. In addition, the filtering and pooling processes were performed using two convolutional layers as the complexity of the dataset was not high.

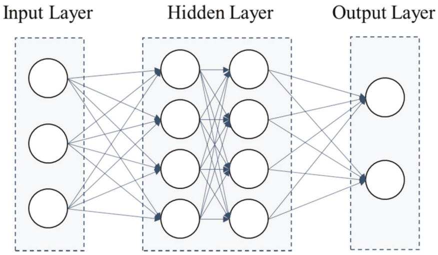


Figure 2. Multilayer perceptron (MLP) process.

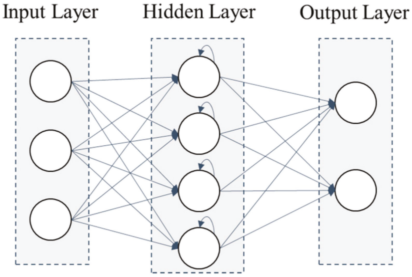


Figure 3. Convolutional neural network (CNN) process.

As described in detail in the experimental results of Section 4, when deep learning is applied to OTT user analysis, the accuracy is higher than when applying general machine-learning methods, although it takes much longer. The number of users of the dataset used in this study is 1,581, which is considerably less than the number of users serviced by ISPs or OTT providers. With larger numbers of simultaneous users, the time consumption could become prohibitive. Therefore, we apply the aforementioned MetaCost framework.

3.2. Time-Aware Consumer Classification Based on MetaCost and Deep Learning

In this study, we classify consumers into three consumption types (i.e., high, low, and average) by analyzing their OTT-usage traffic. Notably, there are two other classes of users that we ignore: Those that use an extremely heavy amount of OTT services and those who rarely use services. As these classes, which have extreme characteristics, can be easily classified via general machine learning, deep learning does not need to be used to classify these extrema. Therefore, we propose a framework that first filters high- and low-consumption consumers through a fast and relatively accurate machine-learning technique. Then, it performs deep-learning-based classification for the remaining customers. This framework shortens the overall computation time by reducing the number of samples to which deep learning is applied, allowing it to focus on the more ambiguous classes [41]. Figure 4 illustrates the proposed time-aware framework based on MetaCost.

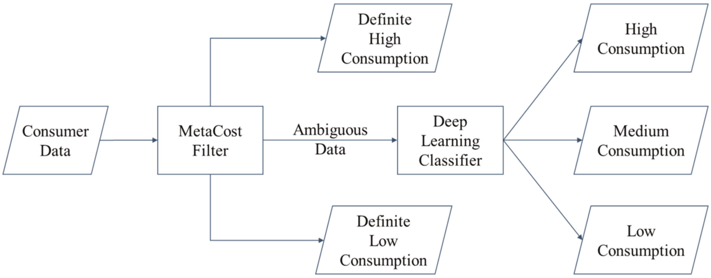


Figure 4. Proposed framework.

Figure 5 shows a simple schematic of the high- and low-consumption filtering process, which forwards the non-extrema data to the deep-learning-based classifier. By setting the cost of errors that misclassify non-high-consumption data as "high" greater than that of errors that incorrectly classify high-consumption data as "medium" or "low", we prevent other classes of data from being misclassified as "definite high-consumption". When classifying definite low-consumption data, we filter out only the obvious data by setting the cost with the contrapositive logic. If the cost is set high in order to not mix the filtered data with other data, all data that are slightly ambiguous are forwarded to the next step, as shown with the "high-cost" process of Figure 5. As a result, the load on deep learning is mitigated. However, there is an increased chance of lower accuracy, because, during the filtering process, medium-consumption data can be misclassified as high- and low-consumption data. We adjust the tradeoff relationship between accuracy and time-consumption by controlling costs according to the number of data and resource state.

In the case of general machine learning, the weights for the errors resulting from the classification process remain the same. The MetaCost method sets the cost of errors differently and is suitable to be applied to the proposed filtering framework, because classification is performed in terms of minimizing costs. The MetaCost method assigns each data to a class satisfying the following equation:

$$x\text{'s class} = \arg\min_{i} \sum\nolimits_{j} p(j|x)\, C(i,j), \tag{6}$$

where $p(j|x)$ is the probability that x belongs to class j, and $C(i,j)$ is the cost incurred when x actually belongs to class j but is classified as class i. After calculating the cost of misclassification for each datum, it is assigned to the class having the lowest cost [41].

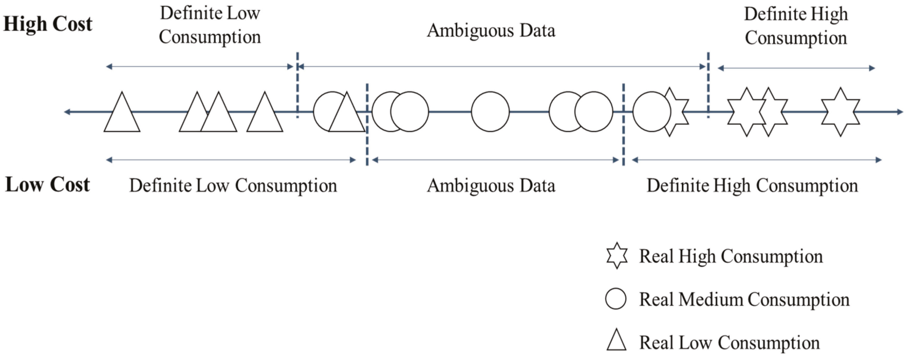


Figure 5. Filtering results according to cost.

The proposed framework's classification time is far less than that of the simple deep-learning method, but deviations can occur based on the cost setting. However, deep learning is generally applied after filtering more than half of the data, making it advantageous over the proposed framework in terms of time consumption. Thus, even with greatly increased sizes of the analysis dataset, the strengths of proposed framework will stand out. As mentioned, the framework increases in flexibility via cost-setting adjustments. If the cost is properly adjusted according to the environment in which this framework is adopted, classification can be performed according to the time and accuracy desired by the analyst.

3.3. Dataset Description

To verify the proposed methodology, we applied the pre-existing dataset mentioned in Section 3.1. Traffic captured directly from the Universidaa del Caucau (Unicauca) network in 2017 was converted into a dataset comprising 130 features and 1581 user samples. These samples were divided into classes of high, medium, and low consumption. The OTT usage data were well represented. Twenty-nine applications were analyzed, including 29 OTT services: Amazon, Apple Store, Apple iCloud, Apple iTunes, Deezer, Dropbox, EasyTaxi, Ebay, Facebook, Gmail, Google suite, Google Maps, HTTP_Connect, HTTP_Download, HTTP_Proxy, Instagram, LastFM, MS OneDrive, Facebook Messenger, Netflix, Skype, Spotify, Teamspeak, Teamviewer, Twitch, Twitter, Waze, WhatsApp, Wikipedia, Yahoo, and YouTube. Features were extracted by analyzing the traffic flow of each service, as shown in Table 1. For each service, features were extrapolated from the dataset [37].

Table 1. Feature Description.

Feature Name	Feature Description
Application-Name.Flows	Amount of the internet-protocol (IP) flow sent by individual users for each OTT service
Application-Name.Flow.Duration.Mean	Average time (s) spent by individual users for being connected to each OTT service
Application-Name.AVG.Packet.Size	Packet size sent by individual users for each OTT service
Application-Name.Flow.Bytes.Per.Sec	Byte size per second sent by individual users for each OTT service

4. Results and Discussion

4.1. Machine and Deep Learning

For classification based on KNN, decision tree, SVM, naïve Bayes, and RIPPER, we employed Weka, a JAVA-based machine-learning library [51]. For MLP and CNN, we employed scikit-learn and TensorFlow. For the experiments, the hardware included an Intel i7-1065G7 processor, 16-GB LPDDR4x memory, and NVIDIA® GeForce® MX250 graphics with GDDR5 2-GB graphic memory. To select machine learning parameters with the best performance for each method, we experimented with adjusting the various parameters to find appropriate values for the OTT datasets. For the KNN, k was set to 17, and J48 was used as the decision tree. In the SVM, a linear kernel was used, and, in RIPPER, the number of folds used for pruning was set to 10. The naïve Bayes classifier used Weka's default settings. For the MLP, ActivationELU was used as the activation function of the hidden and output layers, ADAM was used as the optimizer of the loss function, and AdaDelta was used as the bias updater. For the CNN, two each of convolution, pooling, and fully connected layers were used. ActiviationIdentity and ActivationSoftmax were used as the activation functions of the convolution and output layers, respectively. Adamax was used as the optimizer of the loss function, and AdaDelta was used as the bias updater. Additionally, the number of epochs was set to 100.

We considered recall, precision, and F-measure as the evaluation metrics, calculating them based on the basic true positives (TP), false positives (FP), and false negatives (FN). Recall indicates the number of classes detected among the actual classes, and is the same as TP. Precision is the accuracy of detection and refers to the probability that, when a datapoint is classified into a class, it actually falls into that class. F-Measure is used to obtain the harmonic average value for precision and recall and simultaneously indicates accuracy. These metrics are defined as follows:

$$\begin{gathered}\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}\,,\ \text{Precision} = \frac{\text{TP}}{\text{TP+FP}},\\ \text{F}-\text{Measure} = \frac{2\cdot\text{Recall}\cdot\text{Precision}}{\text{Recall+Precision}}.\end{gathered} \tag{7}$$

Table 2 shows the experimental results based on the aforementioned environment. In the case of the conventional machine-learning methods, KNN achieved good performance with a classification accuracy of 95.1%, and SVM achieved a satisfactory performance of 92.9%. Except for naïve Bayes, all machine-learning methods showed detection rates over 90%, confirming their applicability in classifying OTT users. The accuracy of deep learning was even higher: The use of MLP and CNN to classify consumers achieved a detection rate of 98.2 and 97.6%, respectively. Because the data input was not complex, we observed that the accuracy of MLP was higher than that of CNN. Because both deep-learning methods achieved high performance, we confirmed that their application could also be effectively applied to OTT user analysis.

Table 2. Classification results of machine and deep learning. Abbreviations: k-nearest neighbor (KNN); support-vector machine (SVM); repeated incremental pruning to produce error reduction (RIPPER); multilayer perceptron (MLP); convolutional neural network (CNN).

Machine Learning Algorithm	Recall	Precision	F-Measure	Time (s)
KNN	0.951	0.951	0.950	0.3
Decision tree-J48	0.918	0.918	0.918	0.6
SVM	0.929	0.928	0.928	48.6
Naïve Bayes	0.696	0.726	0.701	0.4
RIPPER	0.908	0.909	0.908	8.1
MLP	0.982	0.982	0.982	298.1
CNN	0.976	0.976	0.976	857.1

Tables 3 and 4 show the detailed classification results of the three types of consumers through deep learning. As shown in Table 3, MLP classified high- and low-consumption users with an accuracy of ≥98%. For medium consumption, although the classification accuracy was lower, it was relatively high at 96%. The results of the CNN shown in Table 4 show a similar tendency. The classification accuracy reached ~99% for high- and low-consumption users, whereas medium consumption showed a relatively accurate detection rate of 94.6%.

Table 3. Detailed classification results obtained through MLP.

	Classified as					
Real Data	High Consumption	Medium Consumption	Low Consumption	Recall	Precision	F-Measure
High Consumption	**98.75%**	1.09%	0.16%	0.988	0.980	0.984
Medium Consumption	2.60%	**96.31%**	1.09%	0.963	0.978	0.970
Low Consumption	0.21%	0.63%	**99.16%**	0.992	0.987	0.989

When classifying OTT users based on deep learning, the classification accuracy was observed to be relatively high, as in other applications fields. However, as confirmed by the classification times shown in Table 2, deep learning took longer to classify consumers than did conventional machine-learning methods. The next subsection describes the MetaCost savings.

Table 4. Detailed classification results of CNN.

	Classified as					
Real Data	**High Consumption**	**Medium Consumption**	**Low Consumption**	**Recall**	**Precision**	**F-Measure**
High Consumption	**98.91%**	1.09%	0%	0.989	0.974	0.981
Medium Consumption	3.46%	**94.60%**	1.94%	0.946	0.973	0.959
Low Consumption	0.21%	1.05%	**98.74%**	0.987	0.981	0.984

4.2. Time-Aware Consumer Classification

To reduce the time required for deep learning, we first classified high- and low-consumption data by using machine learning and MetaCost. We then classified only the remaining ambiguous data using deep learning. KNN and J48 decision trees were the machine learning methods used as the primary filter. Although KNN showed the best performance among all machine-learning methods, J48 achieved fast and highly accurate results. The cost requirement of applying MetaCost is defined as "the cost incurred when classifying data other than high/low consumption as high or low consumption". Therefore, with the cost set to "high", ambiguous data are forwarded to the secondary deep-learning classification. This experiment was performed while changing the cost from 1 to 30 in steps of five. If the cost was one, the weight for all errors was one. Accordingly, the result obtained was the same as that without the application of MetaCost. If the cost was set higher than one, classification was performed to reduce costs. At each step as the cost approached 30, no significant difference was observed from the previous step. Thus, to observe the most conspicuous difference, the experiment was conducted with the cost set to 30. In the secondary classification process, the MLP algorithm was applied for deep learning. Table 5 shows the processing results of the primary filter using KNN and J48 while adjusting the cost from 1 to 30. The table presents the number of data filtered by the primary filter, incorrectly classified by the filter, and processed by deep learning (the secondary classification) with the final detection time.

Table 5. Filtering result according to cost change.

Cost	**Number of Data First Filtered**		**Number of Data Incorrectly Filtered**		**Number of Data Processed by Deep Learning**		**Time Taken for Detection (s)**	
	KNN	**J48**	**KNN**	**J48**	**KNN**	**J48**	**KNN**	**J48**
1	1167	1117	89	65	414	464	71.6	68.7
5	924	1009	14	27	657	572	111.6	88.3
10	792	965	5	13	789	616	145.2	99.5
15	719	950	2	7	862	631	146.9	102.2
20	668	949	2	6	913	632	173.9	102.0
25	633	953	2	19	948	628	177.5	101.8
30	613	958	2	21	968	623	178.7	101.6

With an increase in the cost setting, only the more obvious data were filtered out. Thus, the number of filtered data decreased, and those processed through deep learning increased, resulting in an increase in detection time. However, even if the cost was set to an extremely high value of 30, the detection time was about half. This resulted in the best accuracy while reducing the detection time by more than half. If the cost was set to "low", the detection time was reduced to approximately 23%. However, in this case, the number of incorrect classifications increased, negatively affecting the classification accuracy of the entire framework. When using KNN as the primary filter, results showed fewer errors. However, the number of filtered data was less than that when using J48. Therefore, KNN was determined to utilize more time than J48. On the contrary, although J48 utilized less time because of more filtering, it resulted in more errors. Therefore, the overall classification result of J48 was poor. Table 6 summarizes the overall classification accuracy of the framework per filtering method. Because of space limitations, the detailed results are included in the Appendix A. As shown in the results of Table 6, with an increase in the cost setting, the ambiguous data were forwarded for accurate deep learning, leading to higher accuracy. In terms of accuracy, the results showed only a slight difference when classifying the entire dataset using deep learning. The filter using KNN showed higher accuracy, because it filtered less data than did the filter using J48, resulting in more data being processed during the classification step. Therefore, as observed, the use of the filter with KNN utilized more classification time than that did that of J48. Overall, the filters using KNN and J48 showed classification accuracies of 97 and 96%, respectively, with no significant difference from the value obtained using only deep learning. For both filters, with the cost set higher, the accuracy increased, but the classification time also increased, as shown in Figure 6. Overall, while the accuracy of KNN was high, it utilized more time. When analyzing OTT users, if a considerable amount of data must be analyzed, the focus should be on reducing the time by setting the cost low. Furthermore, if the data to be analyzed are relatively small or if there is sufficient time for analysis, accuracy can be improved by setting the cost high. Thus, optimal time and detection rates can be set while adjusting the cost according to the given environment.

Table 6. Overall classification accuracy of the framework according to cost change.

Cost	KNN Filter + Deep Learning			J48 Filter + Deep Learning		
	Recall	**Precision**	**F-Measure**	**Recall**	**Precision**	**F-Measure**
1	0.930	0.936	0.927	0.945	0.947	0.944
5	0.968	0.968	0.968	0.963	0.963	0.962
10	0.969	0.969	0.969	0.967	0.967	0.967
15	0.972	0.971	0.971	0.967	0.967	0.967
20	0.973	0.973	0.973	0.968	0.968	0.968
25	0.974	0.974	0.974	0.968	0.969	0.968
30	0.976	0.976	0.976	0.968	0.969	0.968

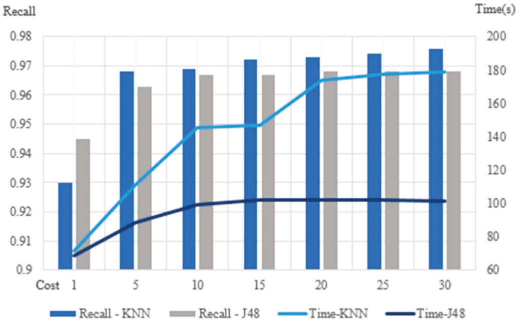


Figure 6. Changes in accuracy and time according to changes in the cost.

The dataset used in this study contained the metadata of 1581 people. Therefore, regardless of analysis time, a significant time difference was not observed. As mentioned, ISPs or OTT providers will likely face hundreds of thousands or millions of users. If the data corresponding to the actual number of users are analyzed using the proposed method, time savings will be clearly observed. To this end, by applying SMOTE [52], an oversampling method, we created a dataset with 159,681 instances. SMOTE is a technique for creating new samples based on existing samples. Unlike other oversampling techniques that simply duplicate existing samples, SMOTE creates synthetic data based on existing data. Therefore, it is possible to create a dataset that has similar characteristics to an existing dataset but has a much larger number of samples. We used SMOTE to create data with a large number of samples, similar to the real environment, and then verified our proposed framework. This amount was approximately 100 times larger than the original dataset. Table 7 shows the time differences between the methods using simple deep learning and the proposed framework based on the oversampled dataset. Because the number of instances grew enormously, the time required for filtering was considerable. However, the time difference was far more conspicuous than that if we had classified the entire dataset using plain deep learning. When analyzing hundreds of thousands or millions of units of data, the proposed framework is confirmed to significantly reduce the time requirements.

Table 7. Changes in time taken for classification according to cost change (unit: Second).

Cost	J48 Filter + Deep Learning			Deep Learning (MLP)
	J48	Deep Learning	Total	
1	568.3	8307.0	8875.3	20369.6
5	2071.2	8397.2	10,468.4	
10	2073.6	8310.3	10,383.9	
15	2102.7	8443.2	10,545.9	
20	2093.1	8303.8	10,396.9	
25	2125.5	8302.0	10,427.5	
30	2091.7	8304.9	10,396.6	

5. Conclusions

In this study, we proposed machine- and deep-learning methods for OTT user analysis to provide ISPs and OTT providers critical timely information about OTT usage data so that they can effectively monitor and execute pricing and mitigation plans. By classifying users according to OTT usage, we confirmed that the classification accuracy was high when using deep learning and conventional machine-learning methods. In particular, deep learning showed higher accuracy. This implies that the application of deep learning to OTT user classification was successful. With plain deep learning, the accuracy of OTT user classification is high, but the classification time takes longer. To shorten this time requirement, we proposed a time-aware MetaCost filtering framework. After first filtering the obvious data using a relatively light algorithm, deep learning was applied to only the most ambiguous data, significantly reducing classification time. However, the accuracy was about the same as with plain deep learning.

This study has the following implications for network and OTT providers. This is the first study that demonstrated how deep learning can be employed to classify OTT user behaviors in a timely manner. ISPs are heavily burdened with applying and maintaining requisite network infrastructure and load balancing to support not only OTT services, but all other internet services, much of which is privately or government contracted. Thus, these investments seriously drive strategy. Hence, timely and extremely accurate usage analysis is needed. This study, therefore, has a wide range of applications in all of those domains.

The proposed framework drastically reduces the time consumption of deep-learning methods with respect to ever-changing user behavior. In fact, when business providers analyze this information, they must consider hundreds of thousands of data items at once. The analysis of such a large amount of data using deep learning can be prohibitively time-consuming and requires heavy computer-resource investments. When applying the proposed method, the costs of time consumption can be drastically reduced.

The proposed method can be used to perform classification according to situations by adjusting the cost factor. In the case where the number of data is relatively small, or there is sufficient time or available resources, accuracy can be improved by increasing the number of data analyzed through deep learning (i.e., cost is set to “high”). On the contrary, if many cases must be analyzed promptly, the cost can be set to “low”. Thus, the more obvious data are filtered out. As such, flexible responses are possible by adjusting the cost factor, and the proposed framework can be, therefore, used by providers for real analysis purposes.

In the future, we plan to focus more on the following points. First, when using deep learning, there is a need for a customized methodology suitable for the particular dataset. Because the OTT dataset used in this study comprised unsophisticated features, a simple MLP or CNN resulted in significant outcomes. However, if complex data were to be analyzed instead, more complex deep learning algorithms must be used. Furthermore, analysis needs to be performed based on various types and categories of OTT user data. To the best of our knowledge, the dataset used in this study is the only public dataset that specializes in OTT. If more datasets related to OTT user behavior will be open to the public in the future, additional and improved research will be possible.

Author Contributions: Conceptualization, J.C. and Y.K.; methodology, J.C.; software, J.C.; validation, J.C. and Y.K.; formal analysis, J.C.; investigation, J.C. and Y.K.; resources, J.C.; data curation, J.C.; writing—original draft preparation, J.C.; writing—review and editing, Y.K.; visualization, J.C. and Y.K.; supervision, J.C. and Y.K.; project administration, J.C. and Y.K.; funding acquisition, Y.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1G1A1099559).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Detailed classification accuracy of each filter according to cost change.

Cost		KNN + Deep Learning			J48 + Deep Learning		
	Filter	**Recall**	**Precision**	**F-Measure**	**Recall**	**Precision**	**F-Measure**
1	1st Filter	0.919	0.920	0.918	0.920	0.920	0.920
	Deep Learning Filter	0.947	0.960	0.951	0.953	0.962	0.956
	Overall	0.930	0.936	0.927	0.945	0.947	0.944
5	1st Filter	0.860	0.898	0.863	0.899	0.913	0.901
	Deep Learning Filter	0.944	0.947	0.945	0.944	0.948	0.945
	Overall	0.968	0.968	0.968	0.963	0.963	0.962
10	1st Filter	0.787	0.874	0.788	0.880	0.905	0.883
	Deep Learning Filter	0.944	0.947	0.945	0.948	0.950	0.949
	Overall	0.969	0.969	0.969	0.967	0.967	0.967
15	1st Filter	0.745	0.862	0.740	0.870	0.899	0.875
	Deep Learning Filter	0.950	0.951	0.951	0.949	0.951	0.950
	Overall	0.972	0.971	0.971	0.967	0.967	0.967
20	1st Filter	0.713	0.854	0.700	0.869	0.899	0.873
	Deep Learning Filter	0.955	0.956	0.955	0.951	0.952	0.951
	Overall	0.973	0.973	0.973	0.968	0.968	0.968
25	1st Filter	0.691	0.848	0.671	0.873	0.901	0.877
	Deep Learning Filter	0.959	0.959	0.959	0.951	0.952	0.951
	Overall	0.974	0.974	0.974	0.968	0.969	0.968
30	1st Filter	0.678	0.845	0.653	0.873	0.901	0.877
	Deep Learning Filter	0.963	0.963	0.963	0.953	0.956	0.954
	Overall	0.976	0.976	0.976	0.968	0.969	0.968

References

1. FCC. Annual Assessment of the Status of Competition in the Market for the Delivery of Video Programming. MB Docket No. 14–16. FCC 15-41. Available online: https://docs.fcc.gov/public/attachments/FCC-15-41A1_Rcd.pdf (accessed on 28 October 2020).
2. MarketsandMarkets. Over-The-Top Services Market by Type (Online Gaming, Music Streaming, VoD and Communication), Monetization Model (Subscription-based, Advertising-based, and Transaction-based), Streaming Device, Vertical, and Region–Global Forecast to 2024. Available online: https://www.marketsandmarkets.com/Market-Reports/over-the-top-ott-market-41276741.html (accessed on 28 October 2020).
3. Statistica. Number of Over-the-top (OTT) Subscription Video Service Subscribers Worldwide from 2012 to 2021. Available online: https://www.statista.com/statistics/821883/number-ott-subscribers/#statisticContainer (accessed on 28 October 2020).
4. Kim, J.; Kim, S.; Nam, C. Competitive dynamics in the Korean video platform market: Traditional pay TV platforms vs. OTT platforms. *Telemat. Inform.* **2016**, *33*, 711–721. [CrossRef]
5. Park, E.A. Business strategies of Korean TV players in the age of over-the-top (OTT) video service. *Int. J. Commun.* **2018**, *12*, 4646–4667.
6. Sujata, J.; Sohag, S.; Tanu, D.; Chintan, D.; Shubham, P.; Sumit, G. Impact of Over the Top (OTT) Services on Telecom Service Providers. *Indian J. Sci. Technol.* **2015**, *8*, 145. [CrossRef]
7. Dai, W.; Baek, J.W.; Jordan, S. Feature article: Network Neutrality [Neutrality between a vertically integrated cable provider and an over-the-top video provider]. *J. Commun. Netw.* **2016**, *18*, 962–974. [CrossRef]
8. Hu, M.; Zhang, M.; Wang, Y. Why do audiences choose to keep watching on live video streaming platforms? An explanation of dual identification framework. *Comput. Hum. Behav.* **2017**, *75*, 594–606. [CrossRef]
9. Rojas, J.S.; Rendon, A.; Corrales, J.C. Consumption Behavior Analysis of Over the Top Services: Incremental Learning or Traditional Methods? *IEEE Access* **2019**, *7*, 136581–136591. [CrossRef]
10. Accenture. The Rise of Cord-Shaving and Cord-Cutting. Available online: http://www.accenture.com/us-en/~{}/media/PDF-30/Accenture-The-Rise-Of-Cord-Shaving-And-Cord-Cutting.pdf (accessed on 28 October 2020).
11. Kim, M.S.; Kim, E.; Hwang, S.; Kim, J.; Kim, S. Willingness to pay for over-the-top services in China and Korea. *Telecommun. Policy* **2017**, *41*, 197–207. [CrossRef]
12. Marvin, R. Netflix, YouTube, Prime Video, and Hulu Dominate Streaming, for Now. Available online: https://www.pcmag.com/news/netflix-youtube-prime-video-and-hulu-dominate-streaming-for-now (accessed on 28 October 2020).
13. Webb, K. Disney Plus Can't Compete with Netflix when it Comes to Original Content, but its Affordable Price and Iconic Franchises Make it a Great Value for Families. Available online: https://www.businessinsider.com/disney-plus-review (accessed on 28 October 2020).
14. Spangler, T. HBO Max and HBO Have 36.3 Million Subscribers, Up 5% From End of 2019, AT&T Says. Available online: https://variety.com/2020/digital/news/hbo-max-subscribers-subscribers-q2-att-1234714316/ (accessed on 28 October 2020).
15. YonhapNews. Local OTT Giant Wavve Sees Drop in Active Users, Netflix Soars: Report. Available online: https://en.yna.co.kr/view/AEN20200617003700320?input=2106m (accessed on 28 October 2020).
16. Kim, J.; Nam, C.; Ryu, M.H. IPTV vs. emerging video services: Dilemma of telcos to upgrade the broadband. *Telecommun. Policy* **2020**, *44*, 101889. [CrossRef]
17. Yoo-chul, K. Netflix May Pay for KT's Network. Available online: http://www.koreatimes.co.kr/www/tech/2020/07/133_293720.html (accessed on 28 October 2020).
18. Johnson, M.R.; Woodcock, J. And Today's Top Donator is: How Live Streamers on Twitch.tv Monetize and Gamify Their Broadcasts. *Soc. Media Soc.* **2019**, *5*. [CrossRef]
19. Floris, A.; Ahmad, A.; Atzori, L. QoE-Aware OTT-ISP Collaboration in Service Management. *ACM Trans. Multimedia Comput. Commun. Appl.* **2018**, *14*, 1–24. [CrossRef]
20. Nevo, A.; Turner, J.L.; Williams, J.W. Usage-based pricing and demand for residential broadband. *Econometrica* **2016**, *84*, 411–443. [CrossRef]
21. Oliveira, T.; Fiorese, A.; Sargento, S. Forecasting Over-the-Top Bandwidth Consumption Applied to Network Operators. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; pp. 00859–00864. [CrossRef]

22. Naudts, B.; Flores, M.; Mijumbi, R.; Verbrugge, S.; Serrat, J.; Colle, D. A dynamic pricing algorithm for a network of virtual resources. *Int. J. Netw. Manag.* **2016**, *27*, e1960. [CrossRef]
23. Mestres, A.; Rodriguez–Natal, A.; Carner, J.; Barlet–Ros, P.; Alarcón, E.; Solé, M.; Hibbett, M.J.; Estrada, G. Knowledge-defined networking. *ACM SIGCOMM Comput. Commun. Rev.* **2017**, *47*, 2–10. [CrossRef]
24. E Middleton, S.; Modafferi, S. Scalable classification of QoS for real-time interactive applications from IP traffic measurements. *Comput. Netw.* **2016**, *107*, 121–132. [CrossRef]
25. Yang, J.; Qiao, Y.; Zhang, X.; He, H.; Liu, F.; Cheng, G. Characterizing User Behavior in Mobile Internet. *IEEE Trans. Emerg. Top. Comput.* **2015**, *3*, 95–106. [CrossRef]
26. Branch, P.; But, J. Rapid and generalized identification of packetized voice traffic flows. In Proceedings of the 37th Annual IEEE Conference on Local Computer Networks, Clearwater, FL, USA, 22–25 October 2012; pp. 85–92. [CrossRef]
27. Bujlow, T.; Riaz, T.; Pedersen, J.M. A method for classification of network traffic based on C5.0 Machine Learning Algorithm. In Proceedings of the 2012 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA, 30 January–2 February 2012; pp. 237–241. [CrossRef]
28. Yuan, R.; Li, Z.; Guan, X.; Xu, L. An SVM-based machine learning method for accurate internet traffic classification. *Inf. Syst. Front.* **2008**, *12*, 149–156. [CrossRef]
29. Shi, H.; Li, H.; Zhang, D.; Cheng, C.; Wu, W. Efficient and robust feature extraction and selection for traffic classification. *Comput. Netw.* **2017**, *119*, 1–16. [CrossRef]
30. Wang, P.; Lin, S.C.; Luo, M. A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs. *IEEE Int. Conf. Serv. Comput.* **2016**, 760–765.
31. Dong, Y.-N.; Zhao, J.-J.; Jin, J. Novel feature selection and classification of Internet video traffic based on a hierarchical scheme. *Comput. Netw.* **2017**, *119*, 102–111. [CrossRef]
32. Yanai, R.B.; Langberg, M.; Peleg, D.; Roditty, L. Realtime Classification for Encrypted Traffic. *Adv. Comput. Intell.* **2010**, 373–385. [CrossRef]
33. Ertam, F.; Avcı, E. A new approach for internet traffic classification: GA-WK-ELM. *Measurement* **2017**, *95*, 135–142. [CrossRef]
34. Davis, J.J.; Foo, E. Automated feature engineering for HTTP tunnel detection. *Comput. Secur.* **2016**, *59*, 166–185. [CrossRef]
35. Zhang, J.; Xiang, Y.; Zhou, W.; Wang, Y. Unsupervised traffic classification using flow statistical properties and IP packet payload. *J. Comput. Syst. Sci.* **2013**, *79*, 573–585. [CrossRef]
36. Du, Y.; Zhang, R. Design of a method for encrypted P2P traffic identification using K-means algorithm. *Telecommun. Syst.* **2013**, *53*, 163–168. [CrossRef]
37. Rojas, J.S.; Gallón, Á.R.; Corrales, J.C. Personalized Service Degradation Policies on OTT Applications Based on the Consumption Behavior of Users. *Multiagent Syst. Technol.* **2018**, 543–557. [CrossRef]
38. Lotfollahi, M.; Siavoshani, M.J.; Zade, R.S.H.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
39. Aceto, G.; Ciuonzo, D.; Montieri, A.; Pescape, A. Mobile Encrypted Traffic Classification Using Deep Learning. In Proceedings of the 2018 Network Traffic Measurement and Analysis Conference (TMA), Vienna, Austria, 26–29 June 2018; pp. 1–8. [CrossRef]
40. Aceto, G.; Ciuonzoa, D.; Montieria, A.; Pescapéab, A. MIMETIC: Mobile encrypted traffic classification using multimodal deep learning. *Comput. Netw.* **2019**, *165*, 106944. [CrossRef]
41. Domingos, P. Metacost: A general method for making classifiers cost-sensitive. In Proceedings of the KDD '99: Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999; pp. 155–164.
42. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K.J. A survey of deep learning-based network anomaly detection. *Clust. Comput.* **2019**, *22*, 949–961. [CrossRef]
43. Rezaei, S.; Liu, X. Deep Learning for Encrypted Traffic Classification: An Overview. *IEEE Commun. Mag.* **2019**, *57*, 76–81. [CrossRef]
44. Kumar, R.; Verma, R. Classification algorithms for data mining: A survey. *Int. J. Innovations Eng. Tech.* **2012**, *1*, 7–14.
45. Kotsiantis, S.B.; Zaharakis, I.; Pintelas, P. Supervised machine learning: A review of classification techniques. Emerging Artif. *Intell. Appl. Comput. Eng.* **2007**, *160*, 3–24.

46. Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
47. Archana, S.; Elangovan, K. Survey of classification techniques in data mining. *Int. J. Compu. Sci. Mob. Appl.* **2014**, *2*, 65–71.
48. Cohen, W.W. Fast Effective Rule Induction. In Proceedings of the International Conference on Machine Learning, Tahoe City, CA, USA, 9–12 July 1995; Elsevier BV: Amsterdam, The Netherlands, 1995; pp. 115–123.
49. Wang, P.; Chen, X.; Ye, F.; Sun, Z. A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning. *IEEE Access* **2019**, *7*, 54024–54033. [CrossRef]
50. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48. [CrossRef]
51. Eibe, F.; Hall, M.A.; Witten, I.H.; Kaufmann, M. *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed.; Morgan Kaufmann: Amsterdam, The Netherlands, 2016.
52. Chawla, N.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Effectiveness of Machine Learning Approaches Towards Credibility Assessment of Crowdfunding Projects for Reliable Recommendations

Wafa Shafqat [1], Yung-Cheol Byun [1,*] and Namje Park [2]

[1] Department of Computer Engineering, Jeju National University, Jeju 63243, Korea; wafashafqat@jejunu.ac.kr
[2] Department of Computer Education, Teachers College, Jeju National University, Jeju 63243, Korea; namjepark@jejunu.ac.kr
* Correspondence: ycb@jejunu.ac.kr

Received: 22 October 2020; Accepted: 16 December 2020; Published: 18 December 2020

Abstract: Recommendation systems aim to decipher user interests, preferences, and behavioral patterns automatically. However, it becomes trickier to make the most trustworthy and reliable recommendation to users, especially when their hardest earned money is at risk. The credibility of the recommendation is of magnificent importance in crowdfunding project recommendations. This research work devises a hybrid machine learning-based approach for credible crowdfunding projects' recommendations by wisely incorporating backers' sentiments and other influential features. The proposed model has four modules: a feature extraction module, a hybrid LDA-LSTM (latent Dirichlet allocation and long short-term memory) based latent topics evaluation module, credibility formulation, and recommendation module. The credibility analysis proffers a process of correlating project creator's proficiency, reviewers' sentiments, and their influence to estimate a project's authenticity level that makes our model robust to unauthentic and untrustworthy projects and profiles. The recommendation module selects projects based on the user's interests with the highest credible scores and recommends them. The proposed recommendation method harnesses numeric data and sentiment expressions linked with comments, backers' preferences, profile data, and the creator's credibility for quantitative examination of several alternative projects. The proposed model's evaluation depicts that credibility assessment based on the hybrid machine learning approach contributes efficient results (with 98% accuracy) than existing recommendation models. We have also evaluated our credibility assessment technique on different categories of the projects, i.e., suspended, canceled, delivered, and never delivered projects, and achieved satisfactory outcomes, i.e., 93%, 84%, 58%, and 93%, projects respectively accurately classify into our desired range of credibility.

Keywords: LDA; LSTM; crowdfunding; project recommendation system; optimization; deep learning

1. Introduction

Recommendation systems aim to assist users in daily decision-making processes and are being utilized by perpetually developing online business ventures. Crowdfunding is a platform that plays the role of a venture capitalist for entrepreneurs with creative minds. The recommendation in crowdfunding becomes trickier and complicated than offline businesses due to many challenges, such as information scrutiny and less proficient investors [1]. Moreover, online data is less reliable and inclined to alteration, making it difficult for investors to rely on a new business idea [2]. Therefore, the credibility assessment of crowdfunding projects becomes an absolute necessity to mitigate the risk of fraud. Crowdfunding is undoubtedly becoming popular as a study [3] shows that approximately 6,445,080 fundraising campaigns were hosted in 2019, with gaming companies being the most successful in generating profit. It is also predicted that the growth of transaction rate annually will reach up to 5.8%, resulting in a total amount of 1180.5 million dollars by 2024 [4]. Despite the remarkable development and inexhaustible possibilities that crowdfunding provides, the challenges and risks of trust, reliability, transparency, etc., are equally daunting, seemingly mounting ones. This study attempts to plug that credibility gap by analyzing and filtering key players' features towards trust-building among investors and the creator. In addition to basic campaign features, we also concentrate on the comments section of the crowdfunding sites, which plays a significant role in fighting against the considerable risks of deceitful online events.

In this paper, we propose a credibility formulation for project recommendations based on a hybrid model. Our proposed architecture has four modules: a text analysis module for project comments, a deep learning module, a credibility estimation module, and a recommendation module. The input data is based on comments-related features and other project-related features. The comment-related features are derived from a comment section of a campaign where users leave their feedback about particular topics; other project-related features include project funding goal, creator's experience, number of images/videos/updates/comments, etc. We perform tokenization, streaming, stop words removal, and data normalization in the data preprocessing layer. In the next step, we perform parameters estimation and topic modeling through latent Dirichlet allocation (LDA). LDA clusters the words with the same meaning in a single topic and is passed to the long short-term memory (LSTM) layer, where the input data consists of the word embeddings, topic embeddings per time-step, and topic distributions.

The LSTM is then trained against new comments to generate sentiments, i.e., positive, negative, and neutral sentiments. As the output from the LSTM layer, we get topic class, accuracy, and project classification. These results are used in the recommendation module to equate and analyze the product's credibility through sentiment score and authenticity calculation. For optimization, we compute the objective function that has both maximization and minimization function. We also formulate the authenticity score and credibility of the project. Through our developed Equation, we get the credibility score and recommended product as our output. We have developed various equations step by step to build the recommendation system considering the critical aspects of positive and negative comments from users and mentioned how authenticity and credibility inter-related for project evaluation are. Our results show that the proposed approach is feasible for all scenarios and achieves high accuracy in recommendation result and authenticity level evaluation and low error rate. The proposed model's evaluation depicts that credibility assessment based on the hybrid machine learning approach contributes efficient results (with 98% accuracy) than existing recommendation models. We have also evaluated our credibility assessment technique on different categories of the projects and achieved satisfactory outcomes. As 95%, 89%, 58%, and 96% of the projects from their respective categories, i.e., suspended, canceled, delivered, and never delivered projects categories were accurately classified into our desired range of credibility.

The rest of this paper includes related works in Section 2, data in Section 3, the proposed method in Section 4, results in Section 5, and conclusion in Section 6.

2. Related Works

In this era of internet and digitalization, an enormous amount of textual data is generated at a high rate. Text data analysis applications are widespread, starting from customer review analysis to extracting and finding a large dataset's hidden meaning. Blei proposes a novel approach to recognize the topics, which ultimately led to sentiments classification, documents classification, and unlocked relatively many assessment prospects for textual data [5]. Topic models are of crucial importance for the illustration of discrete data and are used in different research fields such as medical sciences [6], software engineering [7], geography [8], and political sciences [9], etc. There are many topic modeling techniques; each has its strengths and limitations. The most frequently used approaches include latent semantic analysis (LSA) [10], probabilistic latent semantic analysis (PLSA) [11], latent Dirichlet allocation (LDA) [12], and correlated topic model (CTM) [13].

LSA's primary focus is to generate different representations of texts based on vectors to create semantic content [10,14]. These vector representations are designed to choose related words by computing the similarity among text data. LSA has many applications such as keyword matching, word quality assessment, power collaborative learning, guidance in career choices, making optimal teams [15], reduction of dimensions [16], and identification of research trends [17]. PLSA was introduced to fix the limitations of LSA [18]. It has many implications, including the differentiation of the words with several meanings and clustering of words that share similar contexts [19]. In [20], PSLA is introduced as an aspect model based on a latent variable responsible for linking observations with unseen class variables. In addition to introducing advancements in LSA, PSLA has many other applications, including recommender systems and computer vision [21–23]. LDA model aims to overcome the limitations of LSA and PSLA in capturing the exchangeability of document words.

LDA being an unsupervised approach for topic modeling, has recently become very popular, mainly for topic discovery in a large corpus. In [24], LDA is used for text mining that is based on Bayesian topic models. LDA is also a generative and probabilistic model that attempts to imitate the writing task. Therefore, it attempts to produce a document if a topic is given. There is a variety of LDA based algorithms used in different domains, including author–topic analysis [25], LDA based bioinformatics [26], temporal text mining [27], supervised topic models, and latent co-clustering, etc. In simple words, LDA's fundamental idea is that each document is represented as a mixture of topics. Each topic represents a discrete probability distribution reflecting each word's likelihood to occur in a specific topic. Therefore, a document is described as probability distributions of words in each topic. Certainly, LDA has many applications such as role discovery [28], emotion topic [29], automatic grading of essays [30], and email filtering [31], etc. Biterm topic modeling (BTM) is a topic modeling approach over short texts. These topic modeling methods are becoming a significant job because of the pervasiveness of the short texts available on the internet. BTM is also used to discover discriminative and comprehensible latent topics from short text [32].

Recommendation system (RS) is an intelligent system that suggests items to users that might interest them. Some of the practical example applications of RSs include movie, book, tourist spot recommendations, etc. It is a point of amusement to discover how, "People you may know" feature on Facebook or LinkedIn. In a personalized RS, users get item suggestions based on their past behaviors and social networks-based interpersonal relationships. There are four categories of personalized recommendation systems based on the approach, content-based filtering, collaborative filtering (CF), knowledge-based filtering, and hybrid. A novel clustering method is proposed in [33] that uses the latent class regression model as a baseline model, which considers both the general ratings and textual reviews. In [34], a system that assesses a user's location as an attribute of a recommendation system is proposed. A recommendation method is suggested in [35], which investigates the difference between user feedback to discover a customer's preferences. It considers user ratings and focuses on the sparsity issue of the data. In [36], a CF method is being suggested that uses ratings of different items and feedbacks on various social networks such as Twitter.

A convolutional neural network (CNN) devised by Krizhevsky et al. is referred to as deep CNN [37] that leaned 1000 semantic concepts for training based on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset. Deep CNN proposed by [38] is not suitable for the clothing domain. Therefore, fully connected layers have been included between the seventh and eighth layers to fill the gap between semantics and mid-level features. In [39], the author built a CNN model for the classification of the music genre. This model comprises two convolutional layers, one fully connected layer, and two max-pooling layers. Further, there are ten softmax units with a logistic regression layer to classify the music genre.

Xin Liu et al. [40] used a fusion of matrix factorization and LDA to build a web content-based recommendation model that recommends to the user's fake credibility information to analyze their reaction and improve the model. Schwarz et al. [41] considered measuring webpage popularity, page rank metric, and popularity of a web page to assess a user's web credibility. Studies [42,43] have shown that varied linguistic features, writing styles, and project creators' patterns reveal how communication impacts crowdfunding projects' success. Generally, crowdfunding success is predicted by extracting LDA's semantic features and then by feature selection and data mining [44]. Most of the literature studies are focused on simple embeddings and have not considered using words plus topic embeddings for LSTM training. We have incorporated these embeddings to make more meaningful recommendations that are highly authentic and trustworthy. Moreover, our methodology is novel because we focus on crowdfunding comments to analyze and formulate their impact on the crowdfunding project's credibility. In other sections, we have presented how we have overcome the shortcomings of the literature studies to build a system that considers several factors to recommend credible crowdfunding projects.

3. Proposed Credibility Formulation for Project Recommendation Based on Hybrid Model

This section elaborates the credibility assessment formulation based on learned topics from text and other vital features. The proposed approach uses LDA and LSTM as underlying methods for the credibility assessment process. The overall procedure is divided into multiple tasks as shown in Figure 1, which primarily includes data collection, features selection, text data analysis for topics discovery, topic classification, and formulation for credibility estimation and recommendations. Each task is elaborated separately in the following subsections.

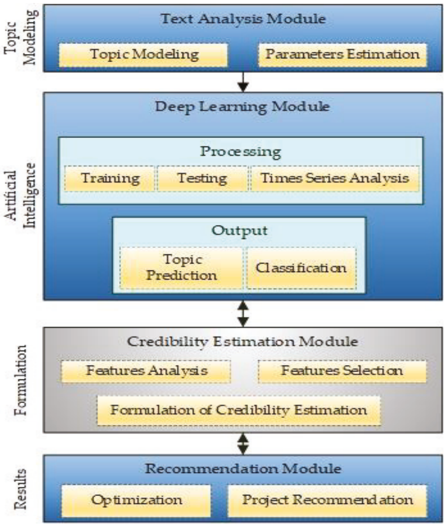


Figure 1. Layered view of the proposed approach.

3.1. Input Data

Each crowdfunding project is rich with the information and data it has in terms of the project's data and user's profile data. The project-based data includes many elements such as project description, duration, number of backers, numbers of comments, and project's success status, etc. Similarly, the user's profile data is related to the creator's information such as name, ID, linked social networks, number of friends, number of created or backed projects, etc. We are mainly focusing on the comments section of a project as comments reveal a lot of information about a project's status and its creator's behavior through backers' experiences. In addition to features extracted from the comment section, we have also focused on the statistical features such as the number of comments, updates, pledged amount, number of backers, etc. We also recorded time delay between different posts to track the project creator's activities. We have collected data from a famous reward-based crowdfunding platform, i.e., Kickstarter. Its mission is to bring creative projects to life that belong to 15 different categories and eight sections: arts, comics & illustration, design and tech, film, food and craft, games, music, and publishing. Table 1 describes the data in detail. There is no limitation on the length of a comment.

Table 1. Input data characteristics.

Data Characteristics	Specifications
Total number of projects	600
Total number of comments (before cleaning)	645,251
Total number of comments (after cleaning)	504,184
Average comments per project	841
Training data	70%
Test data	30%

The temporal patterns of a review, interaction patterns between project backers and creators, the average timeline required from the proposal stage to the approval state varies for every project. The importance of social link and user description in assessing credibility is described in later sections.

3.2. Data Pre-Processing

This unit is in charge of several jobs. It first tokenizes the comments into multiple words. Then these tokenized words are passed through the cleansing unit. Here, all the punctuations are removed, and words are passed through the stemming unit. This unit lower cases all the words and convert each word to its root. (e.g., working is replaced with work). Then, we filter out all the stop words. Stop words are used in any language for grammatical reasons (e.g., a, an, is, etc.) after this processing comment is passed to LDA for further processing.

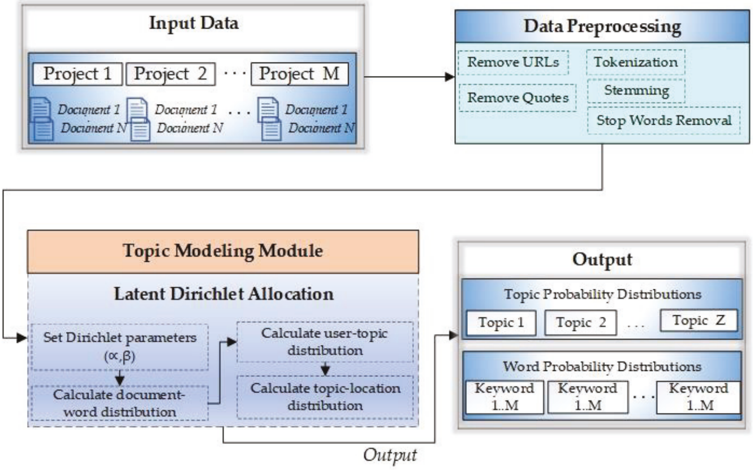
Then we label those clusters into meaningful topics. Therefore, after LDA, we have topic distributions representing the probability of a topic in a document and word distributions representing the probability of a word in a topic. These probability distributions are then prepared as an LSTM input. For LSTM, the word embedding and topic embedding are also generated. These embedding against each new input comment are trained in an LSTM network. The topic classes are distributed in three basic types of sentiments, i.e., positive sentiments, negative sentiments, and neutral. Therefore, the percentage of each topic class is calculated and assigned a sentiment class accordingly.

3.3. LDA and LSTM Based Hybrid Model

The preprocessed data is passed to the hybrid module responsible for the data's primary processing. Here, data is first handed over to the topic modeling process, where LDA is applied. The number of topics and Dirichlet parameters is initiated. LDA generates clusters of words that have the highest similarity.

A. Topic Discovery and Classification

We used LDA for topics discovery in the comments data. We used comments to discover topics as the comments left by backers can present their emotions, feelings, thoughts, and experiences related to the project. Therefore, reviews or comments are powerful enough to shape other's decisions. Figure 2 elaborates on the overall process of LDA. Each project's input data is in the form of comments; each comment is treated as one document that results in N documents per project. Data preprocessing is a crucial and vital part of any NLP technique; therefore, we perform essential yet necessary preprocessing tasks on input data such as removing quotes, stop words, and URLs, tokenization and stemming, etc. Data preprocessing has been influenced by the paper [42], which helps us work with short-texts and proves that LDA works with equivalent efficiency. Once the data is preprocessed, LDA is performed where we set the Dirichlet parameters to calculate desired distributions. We present the output in terms of probability distributions of topics over projects and word distributions over documents. This output is then used as input for the next step, where we use these discovered topics as ground truth and train our LSTM model to predict the topic class of new comments. All the learned topics are divided into different classes, and each class depicts a specific sentiment.



Input Data
Project 1
Project 2
Project M
Document 1
Document N
Data Preprocessing
Remove URLs
Tokenization
Stemming
Remove Quotes
Stop Words Removal
Topic Modeling Module
Latent Dirichlet Allocation
Set Dirichlet parameters (∝,β)
Calculate document-word distribution
Calculate user-topic distribution
Calculate topic-location distribution
Output
Output
Topic Probability Distributions
Topic 1
Topic 2
Topic Z
Word Probability Distributions
Keyword 1..M

Figure 2. Latent Dirichlet allocation (LDA) process.

B. Deep Learning using LSTM

We are using a bidirectional LSTM for capturing the context dependencies concerning time. A bidirectional LSTM is analyzed in its natural order and inverse order when an input is provided to capture maximum dependencies within the data. We are using a 128-unit LSTM (bidirectional) for this purpose. The preprocessing module's input is passed to an embedding layer that converts the input into a 64-bit vector representation. This representation is then processed by the LSTM layer, which is then connected to a dense layer. This layer helps to consolidate the LSTM results. The output layer gives the probability distribution of the output category. The detailed architecture of the proposed approach is presented in Figure 3.

3.4. Project Credibility Estimation

In this section, we present a detailed explanation of the credibility module. The overall process of deriving formulas steps by steps to estimate the credibility of a project is delivered. Trust is an ultimate significant element in any domain that helps to gain the customer's confidence. It is valid for e-commerce sites and online social networks, as well. Therefore, multiple trust-aware recommender systems are being proposed that adopt user's trust statements and their personal or profile data to improve the quality of recommendations considerably.

As we target crowdfunding projects, we aim to formulate an equation to calculate any project's credibility before recommending it to a user. A highly credible recommendation is a project that most likely reflects the user-defined interests and categories with higher chances of its delivery. It must also reflect the lowest probability of factors that can disturb the project's trustworthiness, such as communication delays and less frequent updates, etc. A credible project can precisely be defined as a project with the maximum likelihood of completing and delivering to the backers within the promised period. Various factors are associated with a project's credibility; we define and link a documents' credibility with its estimated authenticity score range. A project's authenticity is a multi-fold view of different and latent aspects, such as latent aspects of a creator's profile and all his or her external social links. It also involves the frequency of account usage and updates from creators. In other words, keeping the backers up to date with each development or progress in the project can earn more credibility points. In addition to that, factors such as the most frequent keywords used, promises related to product delivery or rewards delivery, and investors' sentiments are also crucial. These sentiments of backers are discovered during the LDA process to find latent topics in their comments. There can be multiple topics in a document, and each topic represents a particular class of sentiments. As shown in Table 2 [45], we identified 12 topic classes labeled Topic-1 to Topic-12. The number of topics was varied between 2 and 30 during the experiments to find the optimal number of topics.

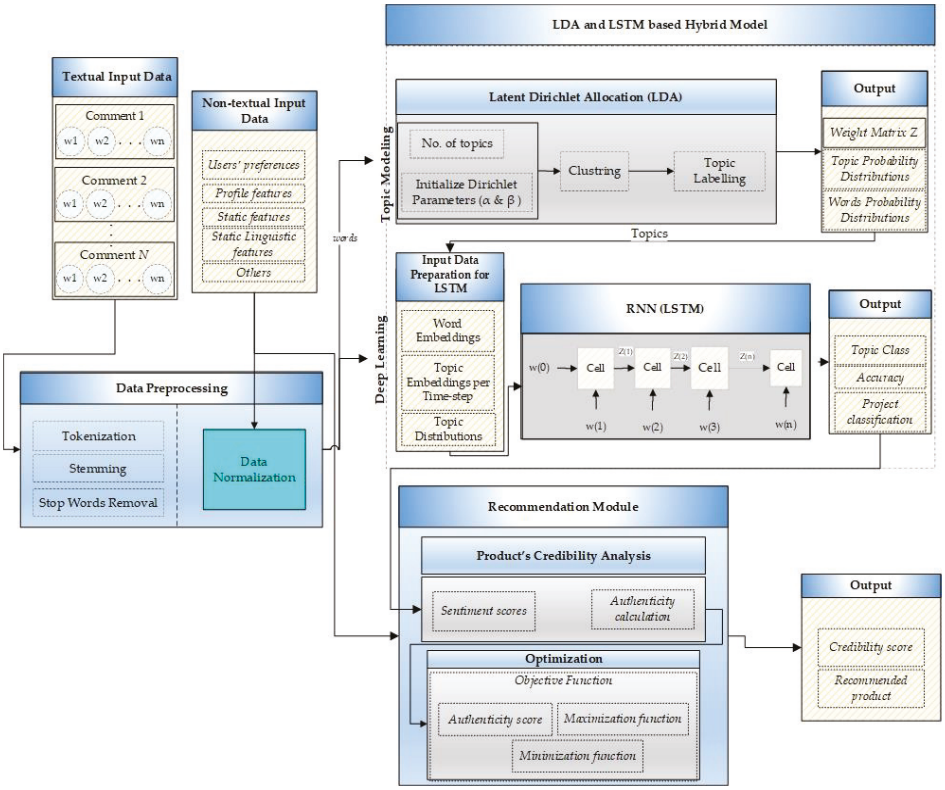


Figure 3. Detailed view of the proposed approach.

The coherence score was increasing as the number of topics was growing. We selected and evaluated the topics based on the coherence score before flattening out, i.e., 12 topics. After training LSTM, the classification of each comment is done into one of these topic classes. We have divided these sentiment classes into three categories, and this division is customized based on the problem, i.e., credibility assessment. These categories are referred to as A, B, and C. Category A is responsible for extremely negative comments, which is represented by Topic-4 to Topic-7; category B means negative reviews, which is characterized by Topic-1 to Topic-3; and category C is representing positive or neutral reviews which are represented by Topic-8 to Topic-12. More emphasis is laid on the negative comments because the negative comments and reviews significantly impact the viewer's mind and decision-making process than positive comments regarding credibility or trust. Therefore, we divided the negative comments into extremely unfavorable class A and negative class B.

Table 2. Topic classes identified using LDA analysis.

Topic Classes	Labels	Popular Words
Topic-1	Waiting for update	Wait, waiting, update, posted, long, silence
Topic-2	Refund inquiry	Return, refund, back, need, please, amount, invoke, demand, reimbursement, want, request, money
Topic-3	Rewards inquiry	Reward, approval, desired, pledge, invoke, request
Topic-4	Legal actions	Legitimacy, filed, report, case, lied, legally, sue, petition, criminal, complaint, signed
Topic-5	Communication gap	Reply, communication, delay, years, last, please, information, progress, since, silent
Topic-6	Product never received	Never, product, receive, deliver, released
Topic-7	Fraud	Ridiculous, Disappointed, Scamming, Scam, fraud, fraudster, ran, product, delete, fail, immoral, thieves
Topic-8	Shipment Information	Ship, delivery, address, time, date, weight, charges, replacement, cancel, received
Topic-9	Product experience	Working, status, advance, easy, difficult, understand, battery, condition
Topic-10	Product description	Weight, color, length, height, memory, soft, dark, light, colorful, single, quantity
Topic-11	Excited about product	Loved, tremendous, excited, awesome, excellent, super, happy, lovely
Topic-12	Product received	Received, fast, today, time, days, quick, late, ago, early, delivered

To evaluate our selected topics, we measured the agreement between two raters using Cohen's Kappa coefficient [46] and followed the process mentioned in [47] to assess our LDA model. Two students (student A and student B) from different laboratories who were unaware of our proposed methodology and had no prior knowledge about the list of LDA topics were requested to extract topics from 250 sampled reviews.

Student A and student B were not allowed to communicate or discuss their thought process behind labeling each review. Student A and student B could identify 9 and 11 topics, respectively. Student A had seven topics in common with LDA, whereas student B had ten topics common with LDA. Among all the topics, we selected six topics that were most common among the two students' topics to measure our LDA model's reliability, as shown in Table 3. As we can see from Table 3, student A and student B have a high degree of agreement for all six topics. The LDA model and respective students' contract is also relatively high, as indicated by the Kappa coefficient.

Category A is for extremely negative comments and severe nature and typically reflects anger by filing lawsuits or complaints. Category B is for relatively simple and generic negative comments that reflect emotions of sadness or disappointment. The classification is based on the nature of malicious content. All other comments belong to category C. The purpose behind this arrangement with more emphasis upon negative comments is the underlying prominence or impact of the malicious content on a product's credibility. Table 4 summarizes the parameters used for authenticity measures with their definitions and notations. In addition to sentiments, we have also included other relevant and impactful features such as readability of content referred to as readScore, the existence of a profile picture, etc.

Table 3. Reliability Assessment of LDA model using Cohen's Kappa coefficient.

Topic Classes	Student A-LDA		Student B-LDA		Student A–Student B	
	Kappa	Overlap	Kappa	Overlap	Kappa	Overlap
Topic-1	0.67 (Moderate)	207 (11)	0.65 (Moderate)	203 (10)	0.69 (Moderate)	223 (22)
Topic-2	0.69 (Moderate)	219 (17)	0.63 (Moderate)	211 (18)	0.79 (Moderate)	234 (24)
Topic-5	0.81 (High)	223 (19)	0.78 (Moderate)	219 (17)	0.85 (High)	239 (25)
Topic-6	0.83 (High)	227 (21)	0.81 (High)	226 (19)	0.89 (High)	238 (21)
Topic-7	0.82 (High)	226 (19)	0.80 (High)	227 (15)	0.87 (High)	230 (20)
Topic-10	0.76 (Moderate)	220 (20)	0.83 (High)	227 (21)	0.85 (High)	236 (19)

Table 4. Definitions of the parameters of authenticity.

No.	Authenticity Parameters	Description	Notations
		Content related Features	
1	Sentiments (-ve)	Percentage of comments that belong to class A	$eNeg$A
2	Sentiments (-ve)	Percentage of comments that belong to class B	$NegB$
3	Sentiments (+ve)	Percentage of comments that belong to class C	$PosC$
4	Readability score	This score reflects the clarity of content, i.e., how easy or difficult it is to understand.	$\text{read}_{\text{Score}}$
		Profile related Features	
5	Profile picture	To check profile picture exists or not	$picY$
6	Social Links	The total number of links (external or social network links, e.g., Facebook, Twitter, etc.),	$\text{Links}_{\text{Ext}}$
7	Communication delay	The time delay between two consecutive posts (update or comment) by the creator	$\text{delay}_{\text{comm}}$

Hence, by incorporating all the factors mentioned above, we have formulated an equation that helps calculate a given project's authenticity. To figure the authenticity of a project, it must first fulfill the eligibility criteria given in Equation (1). Once a project passes the eligibility criteria, Equation (2) is used to calculate the authenticity of it. The eligibility criteria are based on a project's content and partially on the profile associated features in Equation (1).

$$Eligibility_{criteria} = -(eNeg\text{A} + \alpha * picY) \tag{1}$$

Here, α represents the weight associated with the existence of a profile picture. The weightage assigned to $picY$ is lower than the weightage of $eNeg$A because of the level of impact asserted by each parameter. The value of α is set to 0.4. From the above Equation, we define the ranges for both the parameters.

$$eNeg\text{A} = \begin{cases} 0 & if\ A \leq 0 \\ 1 & if\ A > \ 0 \end{cases} \tag{2}$$

Similarly,

$$picY = \begin{cases} 0 & if\ profile\ picture = Exists \\ 1 & if\ profile\ picture = Does\ not\ exists \end{cases} \tag{3}$$

Hence from above Equations (2) and (3), we have

$$Eligibility_{project} = \begin{cases} 0 & favorable \\ < 0 \geq -0.4 & can\ be\ considered \\ < -0.4 & unfavorable \end{cases} \tag{4}$$

Therefore, based on Equation (1) and following the conditions in Equation (4), we can list all possible scenarios of eligibility in Table 5. The content in *eNeg*A is extremely unfavorable as one can sense fears, suspicion, and frustrations in it. Therefore, this category is handled independently to alleviate the probability of any unreliable recommendation. For a reliable project, it must be free from any of the comments in *eNeg*A category. Thus, we used this to set our eligibility criteria. The objective function targets getting the maximum percentage of positive comments, i.e., category C. It also targets to get the maximum number of social links of the project's creator.

Table 5. All possible cases for a project's eligibility criteria.

*eNeg*A	*picY*	Explanation	Eligibility $Eligibility_{project} = -(eNeg\mathrm{A}+\alpha * picY)$ (α = 0.4)
0	0	The author's profile picture exists and no comment belongs to category A of comments	0 (favorable)
0	1	The author's profile picture does not exist and no comment belongs to category A of comments	−0.4 (can be considered)
1	0	The author's profile picture exists and Some comments belong to category A of comments	−1(unfavorable)
1	1	The author's profile picture does not exist and Some comments belong to category A of comments	−1.4 (unfavorable)

In crowdfunding, a backer's faith and confidence rely on the content authenticity and creator's limpidity. Therefore, these aspects are fundamental to a project's success. Table 4 shows that the factor $\mathrm{delay_{comm}}$ is one prime feature of the project, representing a creator's communication styles such as his updates and comments. This feature, $\mathrm{delay_{comm}}$ can be defined as the average time gap between any consecutive posts by the project creator in an update or a comment. It shows the communication rate of a project creator towards the development of a project. Due to the impact of $\mathrm{delay_{comm}}$, the project's authenticity will be damaged if the communication delay upsurges.

After observing and estimating all the relevant features, all the values are normalized between 0 to 1. Here, 0 represents the least authentic feature, and 1 illustrates the highly authentic feature. In other words, these values depict the trustworthiness of a project. Equation (5) below describes this relationship, i.e., the higher the authenticity is, the higher the reliability of a project turns out.

$$Authenticity_{project} \propto Credibility_{project} \tag{5}$$

As a result, a project has different credibility levels, i.e., extremely low, low, normal, high, and extremely high credibility. Each credibility level falls into another degree of authenticity range. The extremely low and low credible projects have higher chances of getting forged. It means the projects with lower credibility levels have the utmost possibilities of fighting with non-payments, no communication or communication delays, delays in posts by the creator in the form of updates or comments, and late or no deliveries. Therefore, such projects are not favorable to be recommended to backers to invest in. Instead, a project with a higher credibility level (high or extremely high credibility) is undoubtedly a profitable project recommended to backers. It has the maximum probability of on-time delivery with more consistent patterns of communication throughout its duration.

For any recommendation system, the percentage of positive and negative reviews is pre-eminent as it reflects a user's attitude towards a product. Therefore, we assess the following points wisely:

1. A fundamental requirement for a product to be reliable is to have a maximum percentage of positive comments and a minimum negative comments rate. A product with a relatively high number of negative reviews becomes less favorable. Therefore, Equations (6) and (7) represent the relationship of comments with authenticity.

$$\text{Authenticity} \propto [\mathit{PosC}\text{i}] \tag{6}$$

and

$$\text{Authenticity} \propto [1/\mathit{NegB}\text{i}] \tag{7}$$

where the percentage of positive and negative comments is referred to as *PosC*i and *NegB*i, respectively.

2. The accessibility of social and profile information such as profile links, display pictures, number of friends or followers, etc., are persuasive and compelling elements for a profile's credibility. Thus, the more a project creator shares personal and relevant information, the easier it gets to earn trust. Therefore, we can say,

$$\text{Authenticity} \propto [\text{Links}_{\text{Ext}}] \tag{8}$$

In the above Equation (8), $\text{Links}_{\text{Ext}}$ is the number of links a person provides for his/her external social media networks, such as Facebook, Twitter, etc.

3. The clarity of speech also plays a vital role in trust development. If the content is easy to follow and understand, a user will easily connect and comprehend it. It helps diminish the misunderstandings, and the confidence level of the reader increases. Therefore,

$$\text{Authenticity} \propto [1/\text{read}_{\text{Score}}] \tag{9}$$

In Equation (9), $\text{read}_{\text{Score}}$ is the readability score of a document. If $\text{read}_{\text{Score}}$ is high, the document is difficult to follow or to understand. The lower the readability score is, the higher probability is to understand it fast.

4. The communication patterns are the key to trust maintenance. A smoother and consistent communication can help people to put their trust in it. If there is no communication from the product creator, it will cause frustration and anger in backers and lose their interests. Therefore, the communication delay should be minimized between the creator's posts.

$$\text{Authenticity} \propto [1/\text{delay}_{\text{comm}}] \tag{10}$$

In Equation (10), delaycomm is the average delay between any successive posts, i.e., comments or updates by the project creator. The higher delays will negatively affect project authenticity.

5. Hence, we can summarize the factors mentioned above as

$$\text{Authenticity} \propto [\mathit{PosC}\text{i}, \text{Links}_{\text{Ext}}] \tag{11}$$

also,

$$\text{Authenticity} \propto [1/\mathit{NegB}\text{i}, \text{delay}_{\text{comm}}, \text{read}_{\text{Score}}] \tag{12}$$

By combining Equations (11) and (12), Equation (13) is formulated as below,

$$\text{Authenticity} \propto [\mathit{PosC}\text{i}, \text{Links}_{\text{Ext}}/\mathit{NegB}\text{i}, \text{delay}_{\text{comm}}, \text{read}_{\text{Score}}] \tag{13}$$

6. We divide the Equation into two parts; the similar factors based on their priority are combined. Hence, Equation (14) combines sentiment-based factors.

$$\text{Authenticity} = [\mathit{Pos}\text{Ci}/\mathit{Neg}B\text{i}] \tag{14}$$

This factor is only associated with product comments. For higher authenticity, *Pos*Ci has to be greater than *NegB*i. We have combined other features related to the product or creator into one Equation as,

$$\text{Authenticity} = [\text{Links}_{\text{Ext}}/\text{read}_{\text{Score}} + \text{delay}_{\text{comm}}] \tag{15}$$

7. Then combine all the factors in one place results into Equation (16) as below,

$$Authenticity_{\ project} = \left[\sum_{i=1}^{n} \frac{\mathit{Pos}\text{Ci}}{\mathit{NegB}\text{i}} + \left(\frac{\text{Links}_{\text{Ext}}}{\text{read}_{\text{Score}} + \text{delay}_{\text{comm}}}\right)\right] \tag{16}$$

8. At the final step, we apply optimizations and formulate our objective functions. We have both maximization and minimization functions. The maximization function maximizes the values for favorable factors, and the minimization function underrates the cost of the least desirable parameters. Hence, we can now formulate the credibility estimation in terms of maximization and minimization functions in Equation (17).

$$Credibility_{project} = \left[\sum_{i=1}^{n} \frac{max\ (\mathit{Pos}\text{Ci})}{\min(\mathit{NegB}\text{i})} + \left(\frac{max(\text{Links}_{\text{Ext}})}{\min(\text{read}_{\text{Score}}) + \min\left(\text{delay}_{\text{comm}}\right)}\right)\right] \tag{17}$$

For the above Equation, we can define the ranges of all the parameters as below in Equations (18)–(22).

$$\mathit{NegB}\text{i} = \begin{cases} 0 & if\ \%age\ of\ negative\ comments = 0 \\ 1 & if\ \%age\ of\ negative\ comments = 100\% \end{cases} \tag{18}$$

$$\mathit{Pos}\text{Ci} = \begin{cases} 0 & if\ \%age\ of\ positve\ comments = 0 \\ 1 & if\ \%age\ of\ positive\ comments = 100\% \end{cases} \tag{19}$$

$$\text{Links}_{\text{Ext}} = \begin{cases} 0 & if\ Number\ of\ links = 0 \\ > 0 \leq 9 & if\ number\ of\ links > 0 \end{cases} \tag{20}$$

The value of $\text{Links}_{\text{Ext}}$ was decided based on the maximum number of external links provided by the project creator. In our case, the maximum number of links a person can provide is considered to be 9. Therefore, $\text{Links}_{\text{Ext}}$ can have any value between 0 and 9.

$$\text{read}_{\text{Score}} = \begin{cases} near\ 1 & Comprehensible\ (easy\ to\ understand) \\ \geq 50 \leq 100 & Incomprehensible\ or\ vague\ (difficult\ to\ understand) \end{cases} \tag{21}$$

$$\text{delay}_{\text{comm}} = [0 - 365\ days] \tag{22}$$

Following Table 6, we can define the maximum and minimum ranges of each parameter.

Table 6. The value ranges for each credibility parameters.

No.	Parameters for Credibility	Maximum Range	Minimum Range
		Content-based	
1	*NegBi*	100	1
2	*Pos*Ci	100	1
3	$\text{read}_{\text{Score}}$	100	1
		Profile-based	
5	$\text{Links}_{\text{Ext}}$	9	0
6	$\text{delay}_{\text{comm}}$	365	0

4. Implementation and Experimental Setup

In this section, we present our implementation environment, along with the experimental setup, in detail. This section also explains the evaluation metrics used for results assessment.

4.1. Experimental Setup

The core system components include Ubuntu 18.04.1 as an operating system (LTS version), 32 Gb memory, and Nvidia GeForce 1080 as a graphics processing unit (GPU). In addition to the core system component, we used python language for development along with Tensorflow API.

4.2. Evaluation Metrics

The performance of our system is measured by using the following evaluation metrics.

1. Accuracy: The accuracy of the model is calculated by using the following formula as shown in Equation (23)

$$Accuracy = 1 - \frac{||Y - \hat{Y}||F}{||Y||F} \tag{23}$$

where Y & $\hat{Y}$ and represent the actual data and predicted data, respectively.

2. Root mean square error (RMSE): The RMSE is calculated using Equation (24).

$$RMSE = \sqrt{\frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}\left(yi^j - \hat{yi}^j\right)^2} \tag{24}$$

where yi^j and $\hat{yi}^j$ are subsets of Y & $\hat{Y}$ and represent the actual data and predicted data at the jth time sample in the ith session, respectively. M is the total time samples, and N is the number of projects. RMSE is precisely used to evaluate the prediction error. The smaller the value of RMSE is, the better is prediction rate or score according to Equation (25).

$$Prediction\ rate\ \propto\ \frac{1}{\text{RMSE}} \tag{25}$$

While accuracy is used to detect predictions' precision, it has an opposite effect than RMSE on the prediction rate, as shown in Equation (26). The higher the value of accuracy is, the better is the prediction rate.

$$Prediction\ rate\ \propto\ Accuracy \tag{26}$$

5. Results

This section presents the results and analysis for crowdfunding project recommendations based on the user's previous interests and credibility. In Section 5.1, we offer a study of the recommendation results for crowdfunding projects. In Section 5.2, we report the accuracy of the proposed model results compared with other models. Table 7 shows the selection criteria for credible projects with different levels of credibility.

Table 7. Selection Criteria for Credibility.

Decision	Credibility Level of a Project	Range of Authenticity (Score)
Highly undesired	Extremely Low	(>0 ≤ 0.2)
Not desired	Low	(>0.2 ≤ 0.4)
Can be considered	Normal	(>0.4 ≤ 0.6)
Desired	High	(>0.6 ≤ 0.8)
Highly Desired	Extremely High	(>0.8 ≤ 1.0)

5.1. *Statistical Analysis of Recommendation Results*

To observe the results of our recommendation module, we used ground truth data. This data includes 100 projects, 55 non-scams, 20 suspended projects, ten canceled projects, and 15 successfully funded projects. Our proposed model works efficiently with high accuracy in both scenarios, i.e., when projects are from the same category or different categories. This dataset exemplifies all possible use case scenarios. We can evaluate how well our recommendation system performs on each type of project in terms of its funding status. The above Figure 4 shows the percentage for each category of crowdfunding projects.

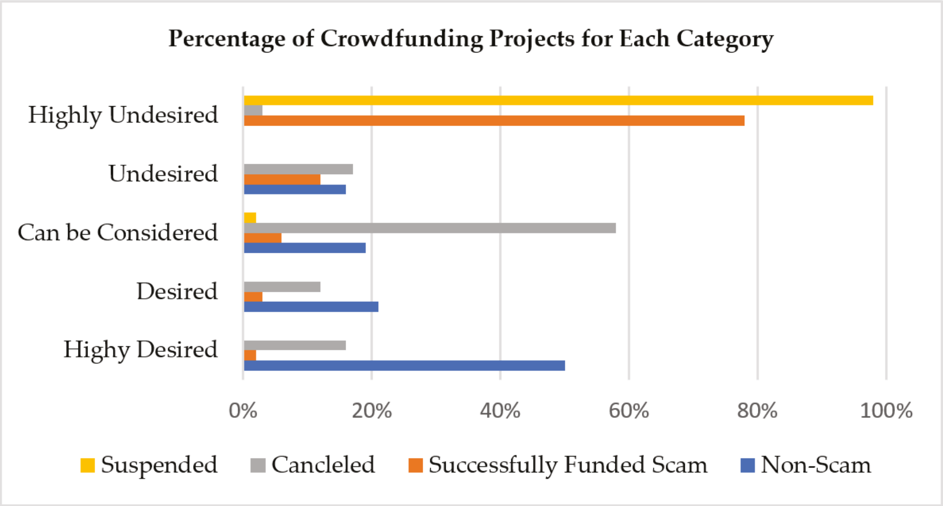


Figure 4. Percentage of crowdfunding projects for each category.

These types are categorized based on the funding status of a project, i.e., "Non-Scam" are projects that are successfully delivered after successful funding; "successfully funded scam" are projects that successfully raised the required funds but failed to deliver; "canceled" category represents projects that have been withdrawn by the project creator before its funding period expires; and "suspended" type means those projects which have been discontinued by the platform in case they figure out any suspicious activity or content.

We have tested our model on all the categories mentioned above of projects to find their authenticity. In Figure 5, we have estimated the authenticity levels for the suspended projects. The *x*-axis shows the authenticity levels between 0 and 1, and the *y*-axis presents the percentage of suspended projects. The estimated authenticity level for 93% of the suspended projects falls in the range (0–0.2).

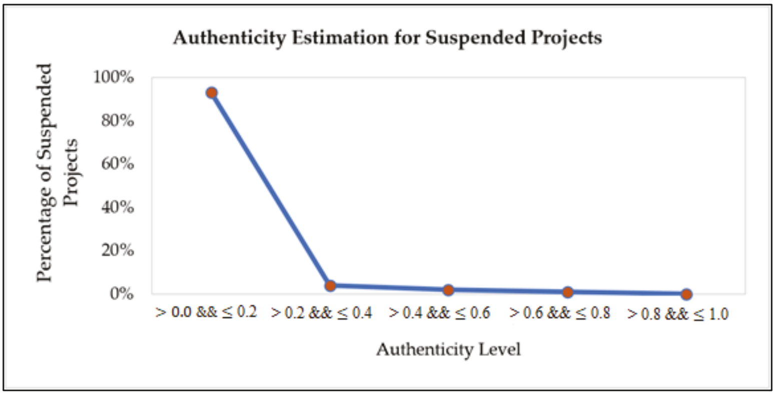


Figure 5. Authenticity estimation for suspended projects.

The data used for this experiment included 80 suspended projects, and 74 projects were falling into the highly undesired range of credibility. This means that these projects are highly undesirable for backers. That is true because these projects are being suspended for some suspicious activities. From this range, we can interpret that most of the projects that have been suspended fail to fulfill the selection criteria of the credibility assessment tool for the recommendation. The statistical analysis of the results is presented in Table 8.

Table 8. Statistical analysis of credibility assessment of suspended projects (Total projects = 80).

Credibility Level	Number of Projects
Highly undesired	74
Not desired	3
Can be considered	2
Desired	1
Highly Desired	0

In Figure 6, we have evaluated the authenticity levels for 70 canceled projects. The estimated authenticity level for 84% of the canceled projects falls in the range (0.4–0.6). Hence, keeping the risk factor in mind, these projects can be considered for investments. The statistical analysis is presented in Table 9. These projects are canceled for multiple reasons, such as lack of funding and budget issues during development phases.

The results show that for most cases, the predicted authenticity level range is between 0 and 0.2. We used 120 undelivered projects, i.e., successfully funded but never delivered projects, and out of these selected projects, 112 projects did not meet the credibility criteria and are highly undesired projects. It represents that regardless of successfully raising funds, backers are disappointed with the progress and development. For such projects, comments play a vital role in understanding a creator's behavior towards his investors after successfully collecting the desired funds. False promises, long delays in communication, or disappearance from the platform are the essential characteristics found in such cases.

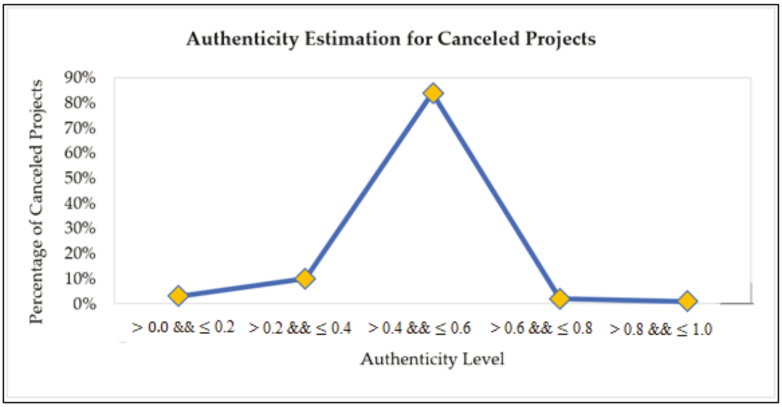


Figure 6. Authenticity estimation for canceled projects.

Table 9. Statistical analysis of credibility assessment of canceled projects (Total projects = 70).

Credibility Level	Number of Projects
Highly undesired	2
Not desired	7
Can be considered	59
Desired	1
Highly Desired	1

Figure 7 presents the accuracy of the recommendation results on the successfully funded projects that didn't deliver, i.e., scam projects.

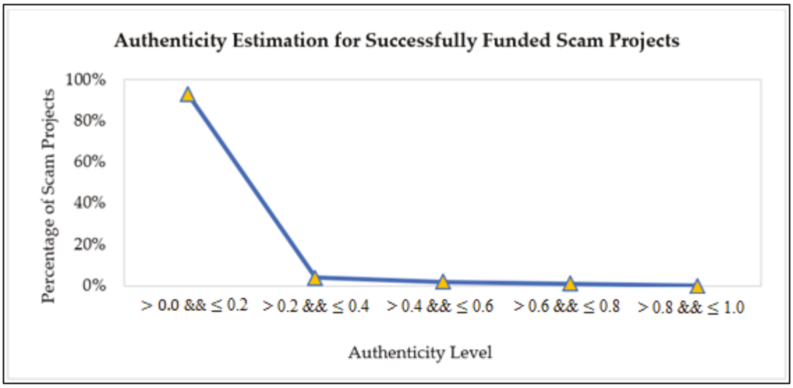


Figure 7. Authenticity estimation for successfully funded scam projects.

The statistical analysis is presented in Table 10. These projects are undelivered and counted as scam projects because the creators didn't fulfill the promises and lacked transparency during the project's development phase.

Table 10. Statistical analysis of credibility assessment of undelivered projects (Total projects = 120).

Credibility Level	Number of Projects
Highly undesired	112
Not desired	5
Can be considered	1
Desired	2
Highly Desired	0

Figure 8 presents an exciting trend. It shows the authenticity level estimation accuracy for non-scam projects.

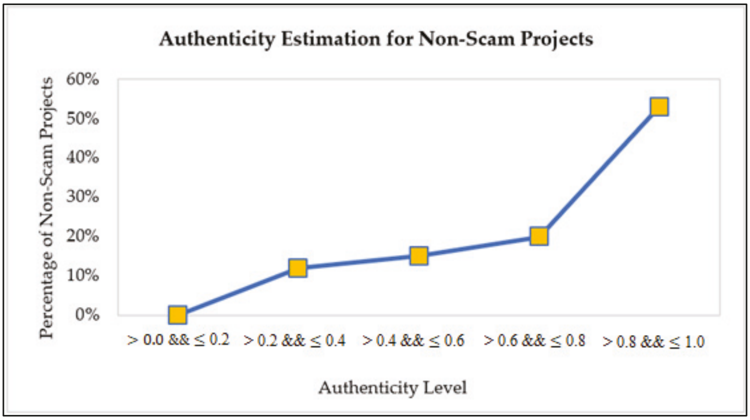


Figure 8. Authenticity estimation for Non-Scam projects.

The inclination depicts rare chances for an authentic and genuine project to have parameters that can lower authenticity scores. Frequently projects are falling in the range of 0.4 to 0.9 and reflecting that comments are going in the gray range, usually for less risky projects.

The statistical analysis is in Table 11. These projects are successfully delivered to the backers. We used 120 successfully delivered projects for this experiment, and 64 of them were highly credible.

Table 11. Statistical analysis of credibility assessment of delivered projects (Total projects = 120).

Credibility Level	Number of Projects
Highly undesired	0
Not desired	14
Can be considered	18
Desired	24
Highly Desired	64

Different learning rates are used for experiments, i.e., 0.1, 0.01, and 0.001 referred to as LR_0.1, LR_0.01, and LR_0.001, respectively, in Figure 9 that evaluate RMSE for a different number of iterations. It can be detected that the testing errors start to get decreased if the learning rate gets smaller. For example, it represents that with a shorter learning rate value, the system's performance improves.

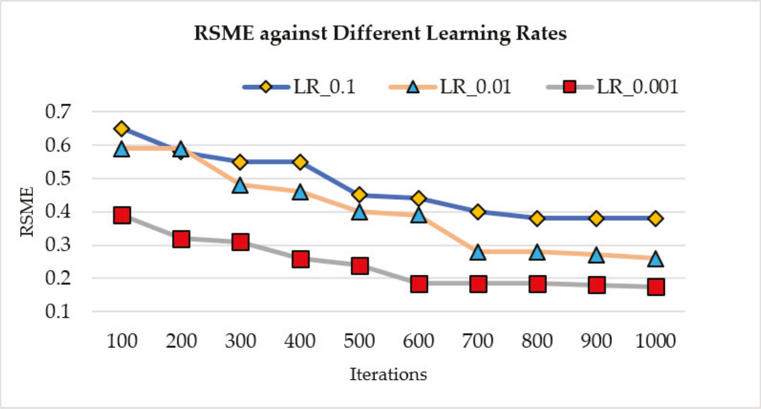


Figure 9. Testing error against different learning rates.

5.2. Comparison with Other Approaches

Here, we have used RMSE as an evaluation metric to evaluate our technique with different ML approaches such as basic NN, bidirectional LSTM, an integrated model of recurrent neural network (RNN) and LDA referred as RNN-LDA, etc. Table 12 presents the RMSE value as a comparison with other models.

Table 12. Evaluation Metrics for Applied Machine Learning Approaches.

ML Approaches	RMSE
Neural Network (NN)	3.37
Bidirectional-LSTM	1.43
RNN-LDA	2.01
NN-LDA	0.82
LDA-LSTM	0.30

Figure 10 presents the accuracy percentage of different models in comparison with our proposed model. It shows that topic models, combined with deep learning models, can achieve better performance than other models.

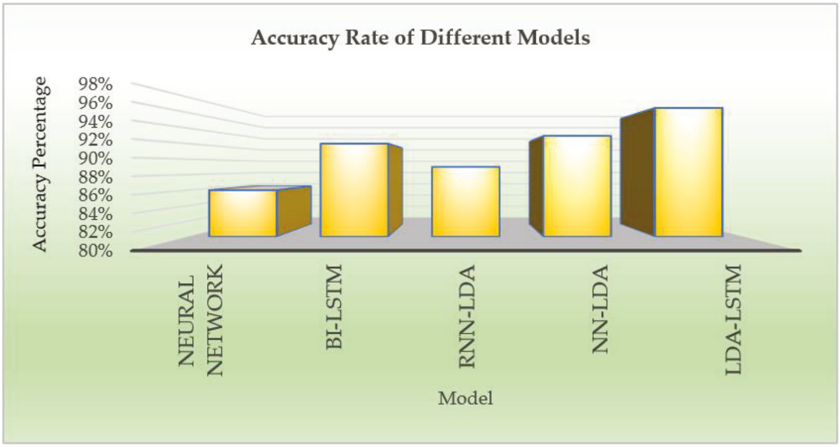


Figure 10. Accuracy Rate of Different Models.

6. Conclusions

We have proposed the methodology for measuring a project's credibility to build a recommendation system. The proposed method uses textual and non-textual data. This system is developed to help the users in selecting reliable and trustworthy options in their preferred categories. The proposed method is a hybrid model of LDA-LSTM and topic modeling that joins the benefits of both (1) LSTMs that captures time dependencies for class and topic prediction and (2) topic modeling that extracts topics that nicely summarize the content. A case study on crowdfunding is performed to analyze and test the proposed system's behavior. We have also embedded an optimized recommendation strategy based on a project's credibility.

This study aims to overcome the limitations of topic models and deep learning and get the most out of both approaches. The main objectives include:

- Finding ways to preserve the contextual dependencies as traditional topic models are based on the bag-of-words approach, so there is a high probability of missing contextual and temporal dependencies.
- Recommendation tools are in use for a long time now; finding the recommended project's credibility is a potential target of this research.

This joint model of LDA-LSTM exploits words and topic embedding, and the temporal data attain 96% accuracy in predicting the topic categories accurately. The topics classes discovered were also evaluated in the context of helping investors identify suspicious campaigns. The prediction quality can be improved if we find out different configurations of comments concerning a project's timeline. We experimented with this by dividing the comments into five various batches of comments. We have not considered projects that have less than 50 comments to maintain the quality of the results.

Many developed applications for recommendation systems in different fields have been proposed. Our proposed approach is a novel approach to recommend a credible crowdfunding project to the best of our knowledge. Moreover, none of the works have focused on crowdfunding comments to find discussion trends and their impact on project credibility. Hence, in crowdfunding, this approach can be used to recommend safe or secure projects to investors. In Table 13, we show a comparison analysis of our proposed model with existing models of recommender systems. In [47–50], the authors use Kickstarter for predictions of the project's success. Others are related to taking comments and updates for estimating the completion of projects and then recommend them to the user. We summarize this research work's contributions: (1) a hybrid method is proposed for reliable and promising recommendations. This approach can model user preferences and word representations in a typical and dynamic style to empower the active measurement of the semantic similarity among the user's preferences and the words. (2) The proposed algorithm is to infer the dynamic embeddings of both the documents and words. We offer a credibility measurement approach for reliable recommendations. The results show that our proposed method outperforms similar state-of-the-art methods significantly.

Table 13. Comparison of our proposed approach with existing recommender systems.

Recommendation Systems	LDA	LDA-LSTM	Language Assessment	Optimization	Comments	Credibility Assessment
[49]	X	X	✓	X	X	X
[50]	✓	X	✓	X	X	X
[51]	✓	X	✓	X	X	X
[52]	✓	X	✓	X	X	X
[53]	X	X	✓	X	X	X
[54]	✓	X	✓	X	✓	X
Proposed Model	✓	✓	✓	✓	✓	✓

7. Discussion

Recommendation systems help users in their decision-making process. Many applications of these systems nowadays in every domain, e.g., location recommendation to tourists, product recommendation to online buyers, restaurant recommendation, route recommendation for travelers, etc. In other words, the need and importance of recommendation systems are not limited to just one platform; crowdfunding is also taking advantage of such applications to make this platform trustworthy for their investors. In this paper, we propose a hybrid model for crowdfunding project recommendations to backers.

The main contribution of this study is we evaluate different features of a campaign to assess its credibility. A credibility assessment is required to build the trust of backers in a campaign. If a backer is partially aware of a campaign's outcome, he can easily decide on investing in it or not. It is essential to build trustworthy recommendation systems, especially when users' hard-earned money is at risk.

We have tried to delve into the details of a campaign and analyze the outcomes of different campaigns based on their funding status. The hybrid model based on topic modeling and deep learning can (1) learn latent topics in comments, (2) to predict the outcome of a project based on the topics discovered so far, and (3) the credibility formulation process carefully evaluates the impact of each feature on the result of a project.

Author Contributions: W.S. conceived the idea for this paper, designed the experiments, wrote the article, assisted in algorithms implementation, and assisted with design and simulation; Y.-C.B. finalized, evaluated proof-read the manuscript, and supervised the work; N.P. did investigation, proof reading and evaluation. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2019S1A5C2A04083374), and also following are the results of a study on the "Leaders in INdustry-university Cooperation+" Project, supported by the Ministry of Education and National Research Foundation of Korea.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alexander, J.E.; Tate, M.A. *Web Wisdom: How to Evaluate and Create Web Page Quality*; L. Erlbaum Associates, Inc.: Hillsdale, NJ, USA, 1999.
2. Grabner-Kräuter, S.; Kaluscha, E.A. Empirical research in online trust: A review and critical assessment. *Int. J. Hum. -Comput. Stud.* **2003**, *58*, 783–812. [CrossRef]
3. Available online: https://www.mordorintelligence.com/industry-reports/crowdfunding-market (accessed on 20 October 2020).
4. Available online: https://www.statista.com/outlook/335/100/crowdfunding/worldwide (accessed on 20 October 2020).
5. Pergola, G.; Gui, L.; He, Y. TDAM: A topic-dependent attention model for sentiment analysis. *Inf. Process. Manag.* **2019**, *56*, 102084. [CrossRef]
6. Karami, A. *Fuzzy Topic Modeling for Medical Corpora*; University of Maryland: Baltimore County, MD, USA, 2015.
7. Asuncion, H.U.; Asuncion, A.U.; Taylor, R.N. Software traceability with topic modeling. In Proceedings of the 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 1–8 May 2010; Volume 1, pp. 95–104.
8. Ghosh, D.; Guha, R. What are we 'tweeting' about obesity? Mapping tweets with topic modeling and Geographic Information System. *Cartogr. Geogr. Inf. Sci.* **2013**, *40*, 90–102. [CrossRef]
9. DiMaggio, P.; Nag, M.; Blei, D. Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of US government arts funding. *Poetics* **2013**, *41*, 570–606. [CrossRef]
10. Dumais, S.T. Latent semantic analysis. *Annu. Rev. Inf. Sci. Technol.* **2004**, *38*, 188–230. [CrossRef]
11. Brants, T.; Chen, F.; Tsochantaridis, I. Topic-based document segmentation with probabilistic latent semantic analysis. In Proceedings of the Eleventh International Conference on Information and Knowledge Management, McLean, VA, USA, 4–9 November 2002; pp. 211–218.

12. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
13. Blei, D.; Lafferty, J. Correlated topic models. *Adv. Neural Inf. Process. Syst.* **2006**, *18*, 147.
14. Landauer, T.K.; Foltz, P.W.; Laham, D. An introduction to latent semantic analysis. *Discourse Process.* **1988**, *25*, 259–284. [CrossRef]
15. Landauer, T.K. Applications of latent semantic analysis. In Proceedings of the Annual Meeting of the Cognitive Science Society, Fairfax, VA, USA, 7–10 August 2002; Volume 24.
16. Sidorov, G. Latent Semantic Analysis (LSA): Reduction of Dimensions. In *Syntactic n-grams in Computational Linguistics*; Springer: Cham, Switzerland, 2019; pp. 17–19.
17. Sehra, S.; Singh, J.; Rai, H. Using latent semantic analysis to identify research trends in openstreetmap. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 195. [CrossRef]
18. Hofmann, T. Probabilistic latent semantic analysis. *Uncertain. Artif. Intell.* **1999**, 289–296. [CrossRef]
19. Hofmann, T. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.* **2001**, *42*, 177–196. [CrossRef]
20. Kakkonen, T.; Myller, N.; Sutinen, E.; Timonen, J. Comparison of Dimension Reduction Methods for Automated Essay Grading. *Educ. Technol. Soc.* **2008**, *11*, 275–288.
21. Liu, S.; Xia, C.; Jiang, X. Efficient Probabilistic Latent Semantic Analysis with Sparsity Control. In Proceedings of the IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13–17 December 2010; pp. 905–910.
22. Romberg, S.; Hörster, E.; Lienhart, R. *Multimodal pLSA on Visual Features and Tags*; The Institute of Electrical and Electronics Engineers Inc.: Cancun, Mexico, 2009; pp. 414–417.
23. Wu, H.; Wang, Y.; Cheng, X. *Incremental Probabilistic Latent Semantic Analysis for Automatic Question Recommendation*; ACM: New York, NY, USA, 2008; pp. 99–106.
24. Shen, Z.Y.; Sun, J.; Shen, Y.D. Collective Latent Dirichlet Allocation. In Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 1019–1025.
25. Rosen-Zvi, M.; Griffiths, T.; Steyvers, M.; Smyth, P. The author-topic model for authors and documents. In Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, Banff, AB, Canada, 7–11 July 2004; pp. 487–494.
26. Liu, L.; Tang, L.; Dong, W.; Yao, S.; Zhou, W. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus* **2016**, *5*, 1608. [CrossRef]
27. Wang, X.; McCallum, A. Topics over time: A non-markov continuous-time model of topical trends. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; pp. 424–433.
28. McCallum, A.; Wang, X.; Corrada-Emmanuel, A. Topic and role discovery in social networks with experiments on enron and academic email. *J. Artif. Intell. Res.* **2007**, *30*, 249–272. [CrossRef]
29. Bao, S.; Xu, S.; Zhang, L.; Yan, R.; Su, Z.; Han, D.; Yu, Y. Joint Emotion-Topic Modeling for Social Affective Text Mining. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, Miami, FL, USA, 6–9 December 2009; pp. 699–704.
30. Kakkonen, T.; Myller, N.; Sutinen, E. Applying latent Dirichlet allocation to automatic essay grading. *Lect. Notes Comput. Sci.* **2006**, *4139*, 110–120.
31. Bergholz, A.; Chang, J.; Paaß, G.; Reichartz, F.; Strobel, S. Improved phishing detection using model-based features. In Proceedings of the CEAS 2008—The Fifth Conference on Email and Anti-Spam, Mountain View, CA, USA, 21–22 August 2008.
32. Cheng, X.; Yan, X.; Lan, Y.; Guo, J. Btm: Topic modeling over short texts. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 2928–2941. [CrossRef]
33. Wang, H.; Lu, Y.; Zhai, C. Latent aspect rating analysis on review text data: A rating regression approach. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 25–28 July 2010; pp. 783–792.
34. Park, M.H.; Hong, J.H.; Cho, S.B. Location-based recommendation system using bayesian user's preference model in mobile devices. In *International Conference on Ubiquitous Intelligence and Computing*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1130–1139.
35. Pazzani, M.J.; Billsus, D. Content-based recommendation systems. In *the Adaptive Web*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 325–341.

36. Greer, C.F.; Ferguson, D.A. Using Twitter for promotion and branding: A content analysis of local television Twitter sites. *J. Broadcasting Electron. Media* **2011**, *55*, 198–214. [CrossRef]
37. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *60*, 84–90. [CrossRef]
38. Lin, K.; Yang, H.F.; Liu, K.H.; Hsiao, J.H.; Chen, C.S. Rapid clothing retrieval via deep learning of binary codes and hierarchical search. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, Shangai, China, 23–26 June 2015; pp. 499–502.
39. Chiliguano, P.; Fazekas, G. Hybrid music recommender using content-based and social information. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016.
40. Liu, X.; Nielek, R.; Adamska, P.; Wierzbicki, A.; Aberer, K. Towards a highly effective and robust Web credibility evaluation system. *Decis. Support Syst.* **2015**, *79*, 99–108. [CrossRef]
41. Schwarz, J.; Meredith, M. Augmenting web pages and search results to support credibility assessment. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC, Canada, 7–12 May 2011; pp. 1245–1254.
42. Mitra, T.; Gilbert, E. The language that gets people to give: Phrases that predict success on kickstarter. In Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, Baltimore, MD, USA, 15–19 February 2014; pp. 49–61.
43. Parhankangas, A.; Renko, M. Linguistic style and crowdfunding success among social and commercial entrepreneurs. *J. Bus. Ventur.* **2017**, *32*, 215–236. [CrossRef]
44. Yuan, H.; Lau, R.Y.; Xu, W. The determinants of crowdfunding success: A semantic text analytics approach. *Decis. Support Syst.* **2016**, *91*, 67–76. [CrossRef]
45. Shafqat, W.; Byun, Y.C. Topic Predictions and Optimized Recommendation Mechanism Based on Integrated Topic Modeling and Deep Neural Networks in Crowdfunding Platforms. *Appl. Sci.* **2019**, *9*, 5496. [CrossRef]
46. Cohen, J. A coefficient of agreement for nominal scales. *Educ. Psychol. Meas.* **1960**, *20*, 37–46. [CrossRef]
47. Jung, Y.; Suh, Y. Mining the voice of employees: A text mining approach to identifying and analyzing job satisfaction factors from online employee reviews. *Decis. Support Syst.* **2019**, *123*, 113074. [CrossRef]
48. Albalawi, R.; Yeap, T.H.; Benyoucef, M. Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis. *Front. Artif. Intell.* **2020**, *3*, 42. [CrossRef]
49. Desai, N.; Gupta, R.; Truong, K. *Plead or Pitch? The Role of Language in Kickstarter Project Success*; Stanford University: Stanford, CA, USA, 2015.
50. Sawhney, K.; Tran, C.; Tuason, R. *Using Language to Predict Kickstarter Success*; Stanford University: Stanford, CA, USA, 2016.
51. Westerlund, M.; Singh, I.; Rajahonka, M.; Leminen, S. Can short-text project summaries predict funding success on crowdfunding platforms? In *Proceedings of the ISPIM Conference Proceedings*; The International Society for Professional Innovation Management (ISPIM): Manchester, UK, 2019; pp. 1–15.
52. Siering, M.; Koch, J.A.; Deokar, A.V. Detecting fraudulent behavior on crowdfunding platforms: The role of linguistic and content-based cues in static and dynamic contexts. *J. Manag. Inf. Syst.* **2016**, *33*, 421–455. [CrossRef]
53. Cumming, D.J.; Hornuf, L.; Karami, M.; Schweizer, D. *Disentangling Crowdfunding from Fraudfunding*; SSRN, 2016; Available online: https://ssrn.com/abstract=2828919 (accessed on 20 October 2020).
54. Tran, T.; Lee, K.; Vo, N.; Choi, H. Identifying on-time reward delivery projects with estimating delivery duration on kickstarter. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Sydney, Australia, 31 July–3 August 2017; ACM: New York, NY, USA, 2017; pp. 250–257.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Prediction of Stock Performance Using Deep Neural Networks

Yanlei Gu [1,*], Takuya Shibukawa [2], Yohei Kondo [2], Shintaro Nagao [2] and Shunsuke Kamijo [3]

[1] College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan
[2] Daiwa Asset Management Co. Ltd., Tokyo 100-6753, Japan; takuya.shibukawa@daiwa-am.co.jp (T.S.); yohei.kondo@daiwa-am.co.jp (Y.K.); nagao.s@daiwa-am.co.jp (S.N.)
[3] Institute of Industrial Science, The University of Tokyo, Tokyo 153-8505, Japan; kamijo@iis.u-tokyo.ac.jp
* Correspondence: guyanlei@fc.ritsumei.ac.jp; Tel.: +81-77-599-3238

Received: 1 October 2020; Accepted: 10 November 2020; Published: 17 November 2020

Abstract: Stock performance prediction is one of the most challenging issues in time series data analysis. Machine learning models have been widely used to predict financial time series during the past decades. Even though automatic trading systems that use Artificial Intelligence (AI) have become a commonplace topic, there are few examples that successfully leverage the proven method invented by human stock traders to build automatic trading systems. This study proposes to build an automatic trading system by integrating AI and the proven method invented by human stock traders. In this study, firstly, the knowledge and experience of the successful stock traders are extracted from their related publications. After that, a Long Short-Term Memory-based deep neural network is developed to use the human stock traders' knowledge in the automatic trading system. In this study, four different strategies are developed for the stock performance prediction and feature selection is performed to achieve the best performance in the classification of good performance stocks. Finally, the proposed deep neural network is trained and evaluated based on the historic data of the Japanese stock market. Experimental results indicate that the proposed ranking-based stock classification considering historical volatility strategy has the best performance in the developed four strategies. This method can achieve about a 20% earning rate per year over the basis of all stocks and has a lower risk than the basis. Comparison experiments also show that the proposed method outperforms conventional methods.

Keywords: deep neural network; stock performance; earning rate; volatility

1. Introduction

Stock performance prediction is one of the most challenging issues in time series data analysis. How to accurately predict stock performance changing is an open question with respect to the financial world and academia field. Stock performance prediction is a difficult task, due to the complexity and dynamic of the markets and many inexplicit, intertwined factors involved. Economic analysts and stock traders are the earliest pioneers who perform the prediction of stock performance. In the past several decades, thousands of books in stock trading have been published.

Many economic analysts and stock traders have studied the historical patterns of financial time series data and have proposed various methods to predict stock performance. In order to achieve a promising performance, most of these methods require careful selection of index variables and finding the sharing features among the distinguished stocks. William J. O'Neil and M. Weinstein are two representatives of successful traders. They summarized their stock trading experience in the publications [1–4]. William J. O'Neil's CAN SLIM method has a huge following, and also performed well in American Association of Individual Investors (AAII)'s implementation of his model [5]. M. Minervini revealed the proven, time-tested trading system he used to achieve triple-digit returns for five consecutive years, averaging 220% per year [6]. Many followers referred to their methodology in stock trading due to their remarkable achievement.

On the other hand, machine learning models, such as Artificial Neural Networks (ANNs) [7–12], Support Vector Regression (SVR) [13–15], Genetic Algorithms (GA) [16], as well as hybrid models [17] have been widely used to predict financial time series during recent decades. In addition, the time-series problem considers Dynamic Time Warping (DTW) that handles scaling and shifting, which is common in the stock market. The recently developed DTW Network is an algorithm candidate for financial time series data processing [18]. Ramos-Requena et al. [19] used the Hurst exponent to measure the correlation and co-movement between two different series. Krollner et al. [20] surveyed papers using machine learning techniques for financial time series forecasting based on technique categories, such as ANN-based, evolutionary and optimization techniques, and multiple/hybrid methods. Cavalcante et al. [21] provided a comprehensive overview of the most important primary studies, which cover techniques such as ANN, SVM, hybrid mechanisms, optimization, and ensemble methods. The surveys indicate that the approaches differ regarding the number and types of variables used in modeling financial behavior; however, there is no consensus on which input variables are the best to be used. In addition, it is important to note that there is no well-established methodology to guide the construction of a successful intelligent trading system. The profit evaluation of the proposed methods when used in real-world applications are generally neglected [21].

Recently, deep learning, as an advanced version of ANN, has attracted attention in the machine learning field because of its high performance in areas such as image recognition and speech recognition. In the field of financial forecasting, a similar new trend considers that a deep neural network has the possibility to increase the accuracy of stock market prediction [22,23]. There are two main deep learning approaches that have been used in stock market prediction: Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). Rout et al. made use of a low complexity RNN for stock market prediction [24]. Pinheiro et al. explored RNN with character-level language model pre-training for both intraday and interday stock market forecasting. The proposed automated trading system that, given the release of news information about a company, predicted changes in stock prices [25]. Li et al. adopted the Long Short-Term Memory (LSTM) neural network, which is an improved version of RNN, and incorporates investor sentiment and market factors to analyze the irrational component of stock price [26]. Nelson et al. studied the usage of LSTM networks to predict future trends of stock prices based on the price history, alongside with technical analysis indicators [27]. Bao et al. presented a deep learning framework where wavelet transforms (WT), stacked autoencoders (SAEs), and LSTM are combined for stock price forecasting [28]. Fischer et al. used deep learning, random forests, gradient-boosted trees, and different ensembles as forecasting methods on all S&P 500 constituents from 1992 to 2015. One key finding in their research is that LSTM networks outperform memory-free classification methods [29]. In order to show accountability to their customers, Nakagawa et al. proposed to approximate and linearize the learned LSTM models by layer-wise relevance propagation [30].

Compared to RNN and LSTM, there are a relatively few examples of applying CNN for stock market prediction. Sezer et al. proposed a novel algorithmic trading model CNN-TA using a 2-D convolutional neural network based on 2-D images converted from financial time series data [31]. Zhou et al. proposed a generic framework employing LSTM and CNN for adversarial training to forecast high-frequency stock market [32]. On the whole, the LSTM network is the most widely used deep learning technology for stock performance prediction.

As mentioned above, automatic trading systems that use Artificial Intelligence (AI) have become a commonplace topic, but there are few examples that successfully leverage the proven method invented by human stock traders to build automatic trading systems. The first contribution of this study is the development of an intelligent trading system by integrating AI and the knowledge of human stock traders. In this study, the important index variables suggested by economic analysts and stock traders are used in a deep neural network to predict future stock performance. The second contribution of this study is the verification of the effectiveness of the knowledge of human stock traders and various investment strategies for constructing a successful intelligent trading system. In this study, what index variables are the most significant, and how to perform the stock performance prediction to maximize earning and minimize the risk of investment are investigated. This study is focused on Japanese stock data to explore a reliable investment algorithm for the Japanese stock market. This also aims to verify whether the method invented based on United State (US) stocks is also effective in Japanese stocks, because the traders William J. O'Neil and M. Minervini summarize their experience based on US stocks.

The rest of the paper is organized as follows: Section 2 describes the important index variables and four strategies for stock classification. Section 3 presents the proposed deep neural network to classify the distinguished stocks with good performance. Section 4 shows the evaluation of the proposed systems. Finally, Section 5 concludes this paper.

2. Important Index Variables and Stock Classification

2.1. Important Index Variables for Stock Performance Prediction

In the current stock market, there are hundreds of index variables indicating the value of a stock from different aspects. Professional analysts and stock traders have tried hard to find the correlation between variables and the future performance of stocks. William J. O'Neil and M. Minervini provided many important points and rules for successful stock trading [1,2]. Table 1 lists the 21 index variables (are also called features) that are the most frequently used for recognizing the distinguished stocks in their related publications [1,2].

Among the most important issues in the development of an intelligent trading system is to decide what features should be used for stock performance prediction. One way is just following the suggestions of human stock traders and feeding all features into the developed system. In addition to using all these suggested features, this study presents a feature selection test and verifies the effectiveness of those features. Based on the definition and characteristic of the features, the 21 features are categorized into four groups: price-related features, trading volume features, company financial status-related features, and others. The results of the feature selection test are discussed in Section 4.

In this study, the related data of the important indices with a weekly resolution are downloaded from the stock database and these important features are used as the input of the deep learning algorithm. Daily price-related data and daily trading volume data are also used as the input data of the deep learning algorithm because these two kinds of data (price and trading volume) are the most important for the prediction of stock price from the viewpoint of human stock traders. Using the additional daily resolution data can avoid missing significant dynamic in each week. When the weekly and daily data are used together, the two kinds of data should be synchronized based on time. The solution for data synchronization is explained in Section 3.

Table 1. Important index variables for stock performance prediction.

Important Index Variables (Features)	Resolution	Group
Price/Earnings Ratio	Weekly	Price-related features
Relative Strength	Weekly	
Price Average of 10 Weeks	Weekly	
Price Average of 30 Weeks	Weekly	
Price Average of 40 Weeks	Weekly	
Year Price High	Weekly	
Year Price Low	Weekly	
Price Open	Weekly + Daily	
Price High	Weekly + Daily	
Price Low	Weekly + Daily	
Price Close	Weekly + Daily	
Trading Volume	Weekly + Daily	Trading volume features
Earnings Per Share 1 Year Growth	Weekly	Company financial status-related features
Earnings Per Share 3 Year Growth	Weekly	
Earnings Per Share 5 Year Growth	Weekly	
Quarterly Profit Growth Rate Relative to the Last Quarter	Weekly	
Quarterly Profit Growth Rate Relative to the Same Quarter in the Last Year	Weekly	
Quarterly Sales Growth Rate Relative to the Last Quarter	Weekly	
Quarterly Sales Growth Rate Relative to the Same Quarter in the Last Year	Weekly	
Common Shares Outstanding	Weekly	Others
Years in the Stock Market	Weekly	

2.2. Definition of Positive Samples for Stock Classification

One of the simplest ways of constructing an intelligent trading system is to employ the binary classification algorithm to classify all stocks as two groups: positive samples which are the stocks with good future performance, and negative samples which are the rest of the stocks. This study presents four strategies for classification, and evaluates the strategies from the aspects of both earning rate and risk of investment.

2.2.1. Constant Threshold-Based Stock Classification

Fund managers are concerned about the rising rate α_C of stock price—expressed in Equation (1). For example, if the price rising rate of a stock could surpass a threshold in the next 12 weeks, this stock could be a good candidate for investment. In Equation (1), CP is the closing price in the current week, and $\mathrm{HP}_{\mathrm{in\_next\_12weeks}}$ denotes the highest price in the next 12 weeks. In this constant threshold-based stock classification, α_C is used to classify stocks. It means that if α_C of a stock is higher than the threshold, the stock will be defined as a positive sample in the stock classification; otherwise, the stock is a negative sample. In this study, 70% was chosen as the threshold based on the experience of fund managers. If the developed intelligent trading system uses the constant threshold-based method to select stocks, the system will simply predict whether α_C of the stock is over the threshold or not. The system will buy the positive stock in the current week, and sell the positive stock when its α_C reaches the threshold in the next 12 weeks.

$$\alpha_C = \frac{\mathrm{HP}_{\mathrm{in\_next\_12weeks}} - \mathrm{CP}}{\mathrm{CP}} \tag{1}$$

2.2.2. Ranking-Based Stock Classification

Because the situation of the market is different every year, in a "good" year, many stocks have good performance and have a high price rising rate. In a "bad" year, the number of stocks with a high price rising rate becomes fewer. In this case, the investment will focus on fewer stocks if the constant threshold method is used for selecting stocks. However, it is necessary to maintain the number of selected stocks and distribute the investment in different stocks to reduce risk.

This study proposes the second strategy in the stock classification. All the stock samples are ranked based on the rising rate α_R expressed in Equation (2), then the top x% samples are defined as positive samples and the rest $(100 - x)$% of the samples are negative samples. In Equation (2), CP is the closing price in the current week, and $\mathrm{CP}_{\mathrm{after\_12weeks}}$ denotes the closing price after 12 weeks. In this ranking-based stock classification, α_R is used to classify the stocks. The value of x was empirically decided as 10 in this research. When the developed intelligent trading system uses this method to select stocks, the system will predict whether the stock is ranked as the top 10% or not, and select the 10% stocks as the positive samples for investment. In this method, there is no constant threshold to decide whether the stocks are in the top 10% or not; therefore, it is impossible to use the strategy of the constant threshold-based method (sell the positive stock when α_C of the stock reaches the threshold) for trading. In this method, the developed intelligent trading system will predict whether the α_R of the stock is in the top 10%, and keep the detected top 10% stocks for 12 weeks before selling them. Therefore, α_R is defined as the price rising rate after 12 weeks, which is different from Equation (1).

$$\alpha_R = \frac{\mathrm{CP}_{\mathrm{after\_12weeks}} - \mathrm{CP}}{\mathrm{CP}} \tag{2}$$

2.2.3. Constant Threshold-Based Stock Classification Considering Historical Volatility

In addition to the price rising rate, volatility is also a factor that needs to be considered in the investment. It is necessary to select the stocks which have both a high price rising rate and low volatility. Therefore, this study proposes the third strategy: constant threshold-based stock classification considering historical volatility. In this method, the target rate β_C is defined as Equation (3).

$$\beta_C = \frac{\alpha_C}{\mathrm{STD}\left(\mathbf{CP}_{\mathrm{in\_past\_12weeks}}/\mathrm{CP}\right)} \tag{3}$$

where α_C can be calculated using Equation (1), and $\mathbf{CP}_{\mathrm{in\_past\_12weeks}}$ is a vector which includes the daily closing price in the past 12 weeks. $\mathrm{STD}\left(\mathbf{CP}_{\mathrm{in\_past\_12weeks}}/\mathrm{CP}\right)$ is the standard deviation of the normalized price in the past 12 weeks. The value of $\mathrm{STD}\left(\mathbf{CP}_{\mathrm{in\_past\_12weeks}}/\mathrm{CP}\right)$ can indicate the historical volatility. In this method, if β_C of a stock is higher than a threshold, the stock will be defined as a positive sample in the stock classification; otherwise, the stock is a negative sample. In this study, 8 was chosen as the threshold based on the experience of fund managers. When the developed intelligent trading system uses this strategy, the system will predict whether β_C of the stock is over the threshold or not. The system will buy the positive stock in the current week, and sell the positive stock when β_C of the stock reaches the threshold in the next 12 weeks.

2.2.4. Ranking Threshold-Based Stock Classification Considering Historical Volatility

Similar to the idea in the second strategy, it is also possible to develop the ranking threshold-based stock classification considering historical volatility. In this method, the target rate β_R is defined as Equation (4).

$$\beta_R = \frac{\alpha_R}{\mathrm{STD}\left[\left(\mathbf{CP}_{\mathrm{in\_past\_12weeks}}\right)/\mathrm{CP}\right]} \tag{4}$$

where α_R can be calculated using Equation (2). The developed intelligent trading system will predict whether β_R of the stock is in the top 10%, and keep the detected top 10% stocks for 12 weeks before selling them.

The designed four strategies are summarized in Table 2. Section 4 presents the performance of the stock trading system developed based on the proposed 4 strategies.

Table 2. Summary of the proposed four strategies for stock classification.

Strategy	Variable Used to Classify Stocks
Constant Threshold-Based Stock Classification	$\alpha_C = \frac{\mathrm{HP}_{\mathrm{in\_next\_12weeks}} - \mathrm{CP}}{\mathrm{CP}}$
Ranking-Based Stock Classification	$\alpha_R = \frac{\mathrm{CP}_{\mathrm{after\_12weeks}} - \mathrm{CP}}{\mathrm{CP}}$
Constant Threshold-Based Stock Classification Considering Historical Volatility	$\beta_C = \frac{\alpha_C}{\mathrm{STD}\left(\mathbf{CP}_{\mathrm{in\_past\_12weeks}}/\mathrm{CP}\right)}$
Ranking Threshold-Based Stock Classification Considering Historical Volatility	$\beta_R = \frac{\alpha_R}{\mathrm{STD}\left(\mathbf{CP}_{\mathrm{in\_past\_12weeks}}/\mathrm{CP}\right)}$

3. Deep Neural Network-Based Model for Stock Performance Prediction

3.1. Long Short-Term Memory Networks

Long Short-Term Memory networks—usually just called “LSTMs”—are a special kind of RNN equipped with a special gating mechanism that controls access to memory cells. Since the introduction of the gates, LSTM and its variant have shown great promise in tackling various sequence modeling tasks in machine learning—e.g., natural language processing, image captioning, and speech recognition. Basically, a LSTM unit consists of an input gate, a forget gate, and an output gate. The architecture of an LSTM unit is shown in Figure 1.

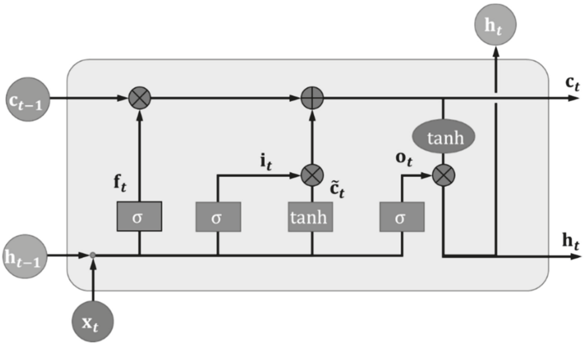


Figure 1. Visualization of a Long Short-Term Memory (LSTM) unit.

Suppose that $\mathbf{x}_t$ is the input and $\mathbf{h}_{t-1}$ is the hidden output from the last time step t-1, the input gate decides how much of the new information will be added to the cell state $\mathbf{c}_t$, and generates a candidate $\tilde{\mathbf{c}}_t$ by:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{5}$$

$$\tilde{\mathbf{c}}_t = \phi(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{6}$$

where $\mathbf{i}_t$ can be thought of as a knob that the LSTM learns to selectively consider $\tilde{\mathbf{c}}_t$ for the current time step. σ is the logistic sigmoid function and ϕ is *tanh*. Generally, **W** terms denote weight matrices (e.g., $\mathbf{W}_{xi}$ is the matrix of weights from the input to the input gate), and **b** terms are the bias vectors. The forget gate decides how previous information will be kept in the new time step, and is defined as:

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right) \tag{7}$$

Then, the cell state $\mathbf{c}_t$ is updated by:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{8}$$

where $\odot$ is the element-wise product of the vectors. Then, the output gate uses the output $\mathbf{o}_t$ to control what is then read from the new cell state $\mathbf{c}_t$ onto the hidden vector $\mathbf{h}_t$ as follows:

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{9}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \phi(\mathbf{c}_t) \tag{10}$$

In this study, the functional LSTM(·,·,·) is used as shorthand for the LSTM model in Equation (11):

$$(\mathbf{h}_t, \mathbf{c}_t) = \mathrm{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{W}, \mathbf{b}) \tag{11}$$

where **W** and **b** include the weight matrices and bias vectors indicated in Equations (5)–(9). The value of **W** and **b** are determined in the training step.

3.2. Concatenated Double-Layered LSTM for Stock Performance Prediction

This study proposes a LSTM-based network to predict the future performance of stocks by classification. The proposed network classifies stocks into two categories (buying or not) based on the historical sequence data. The "Many to one" model has been widely used in sequence data processing. To fully use the memory and forget ability of LSTM, our proposed network is also a "many to one" model. The architecture of the proposed classification network is shown in Figure 2. This means that when classifying the stocks into two categories (buying or not), the historical sequence data from time $t-n$ to t: $(\mathbf{r}_{t-n}, \ldots, \mathbf{r}_t)$ are input into the network together. In the proposed network, n was empirically decided as 52 by considering the experience of professional traders.

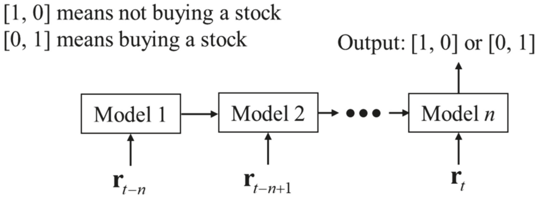


Figure 2. "Many to one" architecture for the classification of stocks.

Figure 3 corresponds to the model in Figure 2. It shows the architecture of the model block. There are double-layered LSTMs. Finally, the output is connected with the last LSTM layer. Here, $\mathbf{x}_t$ denotes the input data at week t. The double-layered LSTM model can be explained by:

$$\left(\mathbf{h}_t^1, \mathbf{c}_t^1\right) = \mathrm{LSTM1}\left(\mathbf{x}_t, \mathbf{h}_{t-1}^1, \mathbf{c}_{t-1}^1, \mathbf{W}^1, \mathbf{b}^1\right) \tag{12}$$

$$\left(\mathbf{h}_t^2, \mathbf{c}_t^2\right) = \mathrm{LSTM2}\left(\mathbf{h}_t^1, \mathbf{h}_{t-1}^2, \mathbf{c}_{t-1}^2, \mathbf{W}^2, \mathbf{b}^2\right) \tag{13}$$

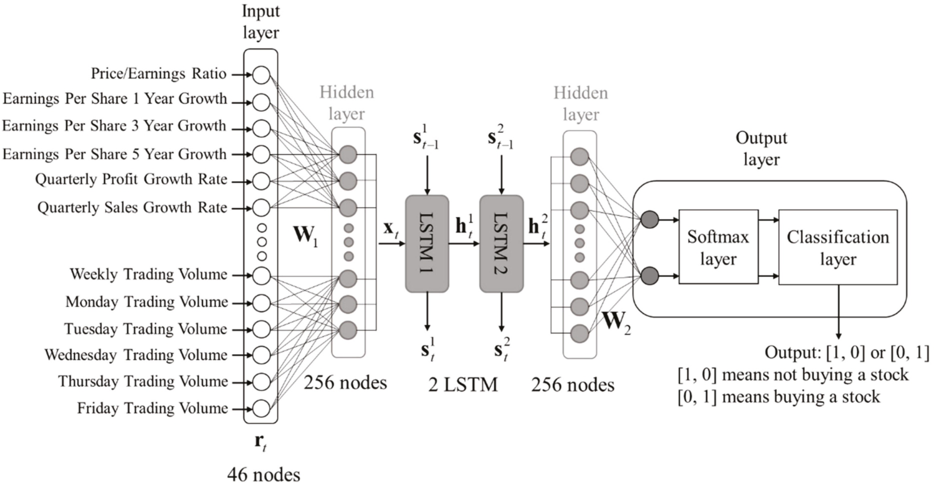


Figure 3. Architecture of double-layer LSTM model for the stock performance prediction.

In Figure 3, $\mathbf{s}_t^n$ stands for $\left(\mathbf{h}_t^n, \mathbf{c}_t^n\right), n = 1 \; or \; 2$. To reduce the complexity of the task, in this study, a binary-class classification system was developed for stock performance prediction. This means that the classification system is expected to recognize two categories. Therefore, the output of the final hidden layer connects two nodes to indicate the probabilities for two categories, the probabilities can be estimated from the output of the second LSTM layer as:

$$\mathbf{P}_t = \mathbf{W}_2 \, \mathbf{h}_t^2 + \mathbf{b}_2 \tag{14}$$

where $\mathbf{W}_2$ is a weight matrix from the hidden layer to the output layer, and $\mathbf{b}_2$ is the bias vector. $\mathbf{P}_t$ is a vector to indicate the probability of the sample for two categories: 0 and 1. The category

"0" means not buying a stock, and the category "1" means buying a stock. The softmax layer and classification layer are responsible for normalization and category selection which are explained in the following equations:

$$\text{Normalization: } \sigma(\mathbf{P}_t)_i = \frac{e^{P_{t,i}}}{\sum_{k=0}^{1} e^{P_{t,k}}} \text{ for } i = 0 \text{ or } 1 \text{ and } \mathbf{P}_t = [P_{t,0}, P_{t,1}\ ] \tag{15}$$

$$\text{Category selection } = \begin{cases} [1,0] & \text{if } \sigma(\mathbf{P}_t)_0 > \sigma(\mathbf{P}_t)_1 \\ [0,1] & \text{else} \end{cases} \tag{16}$$

The double-layer LSTM model can predict whether the automatic trading system should buy or not buy a stock, given the past 52-week history information of that stock. The output of the double-layer LSTM model could be [1, 0] or [0, 1]. [1, 0] means not buying the stock, and [0, 1] means buying the stock. The output ([1, 0] or [0, 1]) is decided in the category selection block based on the comparison of the normalized probabilities provided by the softmax layer. If the probability of category "0" (not buying a stock) is higher than the probability of category "1" (buying a stock), the output is [1, 0]. Otherwise, the output will be [0, 1].

In the training of the network, $\{(\mathbf{r}_{t-n}, \ldots, \mathbf{r}_t); \text{Ground Truth } ([1,0] \text{or} [0,1])\}$ is used as the sample data because of the "many to one" architecture. The training algorithm automatically adjusts the parameters in the model based on the principle of the gradient descent. In the training, the loss function is a cross entropy:

$$Loss = -\sum_{j=1}^{N} \mathbf{T}_j \Big(\log\Big(\sigma\Big(\mathbf{P}_{j,\ t}\Big)\Big)\Big)^T \tag{17}$$

where, N is the number of samples in the training dataset. $\mathbf{T}_j$ is the row vector format ground truth for sample j. The objective of the training process is to minimize the value of Loss Function Equation (17). In Figure 3, both weekly and daily data are used as the input of the LSTM-based deep learning network; one weekly datum can be connected with five daily data from Monday to Friday.

Considering the particularity of the stock classification, the mistakes in the classification have different practical meanings. For example, in comparison to false negative (stocks with good performance are missed in detection), the false positive (stocks are incorrectly detected as good performance stocks) has a higher risk in the real investment. It is possible to give a higher weighting to false positive in the loss function to force the training to reduce the false positive. For example, the ratio of the weighting of false negative and false positive could be 1:2, 1:3, and so on. Therefore, the loss function is reformed as Equation (18). In the experiments, this research presents an attempt to find the best option for parameter $\mathbf{W}_{Loss}$ in order to achieve good performance of the developed trading system.

$$Loss = -\sum_{j=1}^{N} \mathbf{T}_j \Big(\mathbf{W}_{Loss} \odot \log\Big(\sigma\Big(\mathbf{P}_{j,\ t}\Big)\Big)\Big)^T \tag{18}$$

4. Experiment Results

4.1. Experiment Setup and Evaluation Criteria

In this study, a deep neural network was adopted and 52-week historical data of features were used as the input of the deep neural network for a binary classification. For example, when the system performed the classification on 2018/04/01 to select the stocks with good performance in the next 12 weeks, the historical data of 2017/04/03–2018/04/01 (52 weeks) were input into the deep neural network. The ground truth of each historical data is binary data: buying the stock or not, when talking about the future 12-week performance. Because the data are organized weekly, one stock can provide 52 samples per year. For example, the historical data of the samples could be 2017/04/03–2018/04/01, 2017/04/10–2018/04/08, 2017/04/17–2018/04/15, ... , 2018/03/27–2019/03/25. The future 12-week data of these samples are 2018/04/01–2018/06/25, 2018/04/08–2018/07/02, 2018/04/15–2018/07/09, . . . , 2019/03/25–2019/06/17, as shown in Figure 4. In the following description, this paper uses the time of the end of data to denote the 52-week length historical data input into the deep neural network. For example, "2018/04/01" denotes the historical data "2017/04/03–2018/04/01".

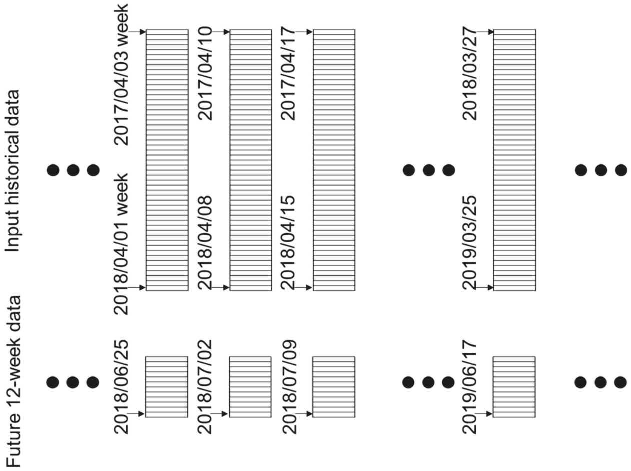


Figure 4. The 52 samples extracted from one stock data.

In this study, the training dataset, validation dataset, and test dataset were separated based on the year and month as shown in Table 3. In order to verify the repeatability of the proposed method, this study presents the evaluation of the different datasets. For example, when the system was tested on the dataset of the period from 2018/04/01 to 2018/09/30 (as indicated in the final row of Table 3), the data from 2017/04/01 to 2017/12/31 were used for validation, and the data from 2001/04/01 to 2016/09/30 were used for training. The datasets in each row of Table 3 are considered as one set. Averagely, the number of samples in each training, validation, and test dataset is about 450,000, 45,000, and 30,000, respectively. It is important to note that there is no overlap among training, validation, and test data in each set. The basic process in the evaluation of each set of datasets is to use the training dataset for training the model and obtaining multiple classifiers. After that, the best classifier is selected based on the validation dataset. Finally, the selected classifier is evaluated in the test dataset. This process was conducted on each set of datasets to demonstrate the repeatability of the proposed methods.

The training dataset was used to train the model. Training is an iteration process with multiple epochs. One epoch means that all training data have been used once for backpropagation. In this study, the number of epochs was set as 50, because 50 epochs are enough for the convergency of the training process. The training process output one classifier after each epoch. Therefore, 50 different classifiers were generated after 50 epochs. Theoretically, the final classifier should have the best performance. However, the performance of the classifiers did not change too much in the training process. One reason is that enough training data were provided for the deep learning algorithm. After several epochs, the training processing converged, and the parameters of the classifier were optimal.

Table 3. Training dataset, validation dataset, and test dataset for repeatability evaluation.

Training Dataset	Validation Dataset	Test Dataset
2001/04/01–2010/03/31 & 2012/10/01–2018/09/30 data	2010/04/01–2010/12/31 data	2011/04/01–2011/09/30 data
2001/04/01–2010/09/30 & 2013/04/01–2018/09/30 data	2010/10/01–2011/06/30 data	2011/10/01–2012/03/31 data
2001/04/01–2011/03/31 & 2013/10/01–2018/09/30 data	2011/04/01–2011/12/31 data	2012/04/01–2012/09/30 data
2001/04/01–2011/09/30 & 2014/04/01–2018/09/30 data	2011/10/01–2012/06/30 data	2012/10/01–2013/03/31 data
2001/04/01–2012/03/31 & 2014/10/01–2018/09/30 data	2012/04/01–2012/12/31 data	2013/04/01–2013/09/30 data
2001/04/01–2012/09/30 & 2015/04/01–2018/09/30 data	2012/10/01–2013/06/30 data	2013/10/01–2014/03/31 data
2001/04/01–2013/03/31 & 2015/10/01–2018/09/30 data	2013/04/01–2013/12/31 data	2014/04/01–2014/09/30 data
2001/04/01–2013/09/30 & 2016/04/01–2018/09/30 data	2013/10/01–2014/06/30 data	2014/10/01–2015/03/31 data
2001/04/01–2014/03/31 & 2016/10/01–2018/09/30 data	2014/04/01–2014/12/31 data	2015/04/01–2015/09/30 data
2001/04/01–2014/09/30 & 2017/04/01–2018/09/30 data	2014/10/01–2015/06/30 data	2015/10/01–2016/03/31 data
2001/04/01–2015/03/31 & 2017/10/01–2018/09/30 data	2015/04/01–2015/12/31 data	2016/04/01–2016/09/30 data
2001/04/01–2015/09/30 & 2018/04/01–2018/09/30 data	2015/10/01–2016/06/30 data	2016/10/01–2017/03/31 data
2001/04/01–2016/03/31 data	2016/04/01–2016/12/31 data	2017/04/01–2017/09/30 data
2001/04/01–2016/09/30 data	2016/10/01–2017/06/30 data	2017/10/01–2018/03/31 data
2001/04/01–2017/03/31 data	2017/04/01–2017/12/31 data	2018/04/01–2018/09/30 data

However, how to choose the best classifier is a problem. In this study, a validation dataset was used to choose the best classifier. As shown in Table 3, the validation dataset is the most recent year before the test dataset. In addition, there is three-month gap between the validation data and test data. When the system works on the day of 2018/03/31 and predicts the future of stocks in the next 12 weeks, the validation dataset should be the data from 2017/04/01 to 2018/03/31. However, the future 12-week data for the historical data from 2018/01/01 to 2018/03/31 are not available on the day 2018/03/31. Therefore, the data from 2018/01/01 to 2018/03/31 cannot be used for validation, and the validation dataset has a 9-month period.

In addition, as described in Section 3, the low false positive value is also expected in the stock selection. Therefore, when choosing the classifier, it is also necessary to consider which one has a low false positive value and maintain the high true positive value at the same time. In this study, the following best precision criterion was adopted to select the classifier in the validation dataset:

$$\underset{\mathrm{i}=1,\ldots 50}{\mathrm{Argmax}}(\mathrm{Precision}),\ \text{where Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{19}$$

where TP is the number of true positive samples and FP is the number of false positive samples.

In the repeatability evaluation, the test dataset had half a year period, and the validation dataset had a 9-month period. The training dataset is the data excluding the test and validation dataset. For example, when the test dataset is data from 2011/04/01 to 2011/09/30, the validation dataset is data from 2010/04/01 to 2010/12/31. In this case, the training dataset was the data from 2001/04/01 to 2010/03/31 and the data from 2012/10/01 to 2018/09/30. In this study, future data after the test data were used for training, because the deep learning needs huge training data to achieve good performance. Using the data after the test data period increases the number of training samples. It is important to note that there is a one-year gap from the end of the test data period to the training data period, because excluding the data in that one year can strictly guarantee that any part of the test data is not used in the training.

In this study, True Positive Rate (TPR, Recall), True Negative Rate (TNR), Average Correction Rate (ACR), and Precision were used to evaluate the performance of the developed prediction systems. The evaluation criteria are denoted in Equations (20)–(23):

$$\mathrm{TPR} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{20}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} \tag{21}$$

$$\mathrm{ACR} = \frac{\mathrm{TPR}+\mathrm{TNR}}{2} \tag{22}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{23}$$

where P is the number of all positive samples, and N is the number of all negative samples. TP is the number of true positive samples, FP is the number of false positive samples, TN is the number of true negative samples, and FN is the number of false negative samples.

In addition to the four criteria, the average maximum price rising rate of stocks and average maximum price decreasing rate of stocks were also used in the evaluation. Moreover, this study also presents a simulation of stock trading to evaluate the performance of the proposed methods, and the details of the simulation are presented in each following each subsection.

4.2. Results of Constant Threshold-Based Stock Classification

In the evaluation of the method of constant threshold-based stock classification, two factors should be discussed: $\mathbf{W}_{Loss}$ parameter in Equation (18), and features in Table 1. Table 4 shows the performance of the classification using all features and $\mathbf{W}_{Loss}$ (1:1) values.

The first column indicates the time period of the test dataset. True negative rate, true positive rate (recall), average correction rate, and precision are listed from the second to fifth columns. The average of the maximum rising rate of detected good performance stocks and all stocks are demonstrated in the sixth and seventh columns. The average of the maximum decreasing rate of detected good performance stocks and all stocks are shown in the eighth and ninth columns.

Moreover, this study also presents a simulation of a real stock trading system. In the case of the binary-class classification system using 70% rising rate threshold, the system sets up a 70% rising rate as the selling point. The system will firstly buy all selected stocks. If a selected stock (detected positive sample) achieves 70% rising rate, the system sells it immediately. Otherwise, the stock is kept and sold by the end of 12 weeks. The tenth and eleventh columns of Table 4 show the earning rate of the simulated stock trading system. In addition, the twelfth column of Table 4 provides the basis of all stocks. The basis is the average earning rate from present to 12 weeks later.

In addition, the other two criteria are used in the evaluation of risk: Sharpe ratio with trading on selling point and Sharpe ratio without trading on selling point. The two criteria are defined as Equations (24) and (25):

$$\text{SR}_\text{T} = \frac{\left(\text{P}_\text{selling} - \text{CP}\right)/\text{CP}}{\text{STD}\left(\mathbf{CP}_\text{until\_selling}/\text{CP}\right)} \tag{24}$$

$$\text{SR}_\text{NT} = \frac{(\text{CP}_\text{after\_12weeks} - \text{CP})/\text{CP}}{\text{STD}(\mathbf{CP}_\text{in\_next\_12weeks}/\text{CP})} \tag{25}$$

where CP is the closing price in the current week, P_selling is the price when selling the stock, and $\text{CP}_\text{after\_12weeks}$ is the close price after 12 weeks. In Equation (24), $\mathbf{CP}_\text{until\_selling}$ is a vector which includes the daily closing price from the current week to selling. $\mathbf{CP}_\text{in\_next\_12weeks}$ is a vector which includes the daily closing price in the next 12 weeks.

In fact, the Sharpe ratio without trading on selling point means the stocks will be sold at the end of 12 weeks. The thirteenth and fourteenth columns of Table 4 show the Sharpe ratio with trading on selling point, the fifteenth and sixteenth columns of Table 4 illustrate the Sharpe ratio without trading on selling point. In addition, the average of all tests is listed in the last row of Table 4. This study presents the evaluation of different features and $\mathbf{W}_{Loss}$ values. Because of the limitation of the page length, Table 5 shows the summary of these evaluations. In this study, the results generated using different $\mathbf{W}_{Loss}$ values were compared, and then the best $\mathbf{W}_{Loss}$ values were chosen for the feature selection. The following conclusions can be obtained from the data in Table 5:

(1) Values of parameter $\mathbf{W}_{Loss}$ affect the earning rate.

(2) Price features can provide the best performance compared to all features.

(3) The best earning rate happens when the stock classification uses $\mathbf{W}_{Loss}$ (1:2), price features. A 2.146% (6.037–3.891%) earning rate per 12 weeks above basis is achieved in constant threshold-based stock classification.

(4) The Sharpe ratio of the detected stock is lower than all stocks, which indicates that the constant threshold-based stock classification method has a relatively high risk in stock classification. The reason for this is that the stock volatility is not considered in this method.

4.3. Results of Ranking-Based Stock Classification

Similar to the constant threshold-based stock classification, this study also presents multiple evaluations for the ranking-based stock classification. In the evaluation, the effect of different values of $\mathbf{W}_{Loss}$ and input features was tested. In the ranking-based stock classification, the top 10% stocks were considered as positive samples. Table 6 shows results generated using the different configurations. There was no fixed threshold for stock trading; therefore, the simulated stock trading system kept the detected or all stocks until the end of the following 12 weeks and then sold them. Thus, there was no Sharpe ratio with trading. The following conclusions can be obtained from the data in Table 6.

(1) Values of parameter $\mathbf{W}_{Loss}$ affect the earning rate.

(2) Price and trading volume-related features can provide the best performance compared to all features.

(3) The best earning rate happens when the stock classification uses $\mathbf{W}_{Loss}$ (1:2), price and trading volume-related features. A 6.984% (10.875–3.891%) earning per 12 weeks above basis is achieved in the ranking-based stock classification.

(4) The Sharpe ratio of the detected stock is lower than all stocks, which indicates the ranking-based stock classification method has a relatively high risk for the stock classification. The reason is that the stock volatility is not considered in this method.

(5) From the aspect of the earning rate, the ranking-based stock classification method is more effective than the constant threshold-based stock classification method.

4.4. Results of Constant Threshold-Based Stock Classification Considering Historical Volatility

Similar to the constant threshold-based stock classification, this study also presents multiple evaluations for the constant threshold-based stock classification considering historical volatility. In the evaluation, the effect of different values of the parameter $\mathbf{W}_{Loss}$ and input features was tested. Table 7 shows results using different configurations. In the ranking-based stock classification, the threshold for target rate β_C was set as eight to classify positive samples. In the simulated trading system, the target rate eight was set as the selling point. If the stock achieves the target rate eight, it will be sold immediately. Otherwise, the system will keep the stock and sell it by the end of 12 weeks. The following conclusion can be obtained from the data in Table 7.

(1) Values of parameter $\mathbf{W}_{Loss}$ affect the earning rate.

(2) Price and trading volume-related features can provide the best performance compared to all features.

(3) The best earning rate happens when the stock classification uses $\mathbf{W}_{Loss}$ (1:1), price and trading volume-related features. A 2.504% (6.395–3.891%) earning per 12 weeks above basis is achieved.

(4) The Sharpe ratio of the detected stock is higher than all stocks, which indicates this method can also reduce risk in stock selection. The reason is that stock volatility is considered in this method.

(5) From the aspects of Sharpe ratio, this method has better performance than the stock classification method without considering historical volatility.

Table 4. Constant threshold-based stock classification with threshold 70%, $\mathbf{W}_{Loss}$ (1:1), all features.

Test Datasets	True Negative Rate	True Positive Rate (Recall)	Average Correction Rate	Precision	Average of Maximum Rising Rate of Stocks Detected as Positive Samples and All (%)		Average of Maximum Decreasing Rate of Stocks Detected as Positive Samples and All (%)		Earning Rate of a Simulated System for Stocks Detected as Positive Samples and All (%)		Basis of All Stocks (%)	Sharpe Ratio$_{T}$ of a Simulated System for Stocks Detected as Positive Samples and All		Sharpe Ratio$_{NT}$ of a Simulated System for Stocks Detected as Positive Samples and All	
					Detected	All	Detected	All	Detected	All		Detected	All	Detected	All
2011/4–2011/9	0.830	0.700	0.765	0.011	10.989	8.727	−11.473	−9.822	−1.998	−1.608	−1.644	−0.573	−0.406	−0.591	−0.409
2011/10–2012/3	0.812	0.631	0.721	0.026	17.994	12.937	−7.874	−6.933	5.857	3.990	3.958	0.694	0.650	0.661	0.640
2012/4–2012/9	0.756	0.745	0.751	0.017	11.552	9.363	−9.112	−8.652	1.439	1.255	1.224	0.047	0.161	0.021	0.153
2012/10–2013/3	0.798	0.506	0.652	0.111	36.373	27.539	−5.689	−4.754	20.129	17.138	17.489	1.974	2.022	1.840	1.971
2013/4–2013/9	0.744	0.765	0.755	0.051	24.319	15.806	−11.492	−9.092	5.680	3.909	4.313	0.449	0.501	0.370	0.475
2013/10–2014/3	0.742	0.683	0.713	0.025	16.142	11.876	−11.484	−8.840	1.241	1.742	1.703	−0.124	0.167	−0.164	0.154
2014/4–2014/9	0.807	0.701	0.754	0.040	19.919	14.266	−7.720	−6.402	7.163	6.158	5.969	0.777	0.978	0.704	0.958
2014/10–2015/3	0.901	0.292	0.597	0.034	19.304	16.012	−6.469	−5.086	7.258	8.198	8.131	0.774	1.270	0.727	1.254
2015/4–2015/9	0.866	0.549	0.707	0.031	18.080	12.300	−14.738	−11.079	0.393	0.749	0.682	−0.339	−0.099	−0.383	−0.109
2015/10–2016/3	0.893	0.683	0.788	0.056	21.651	11.178	−15.699	−13.246	1.413	−2.882	−2.826	−0.361	−0.686	−0.439	−0.698
2016/4–2016/9	0.853	0.654	0.754	0.044	20.150	13.867	−9.798	−8.203	6.169	4.995	4.954	0.557	0.778	0.504	0.765
2016/10–2017/3	0.857	0.560	0.709	0.041	20.610	13.967	−8.299	−6.322	8.911	6.572	6.500	0.838	0.966	0.785	0.954
2017/4–2017/9	0.886	0.598	0.742	0.066	26.583	16.118	−7.371	−4.621	11.945	8.507	8.542	1.126	1.378	1.047	1.363
2017/10–2018/3	0.896	0.552	0.724	0.057	24.044	13.413	−13.861	−9.032	4.193	2.102	2.082	−0.043	0.124	−0.118	0.110
2018/4–2018/9	0.899	0.592	0.746	0.031	16.587	10.434	−14.874	−11.103	−3.138	−2.909	−2.716	−0.895	−0.746	−0.942	−0.753
Average	0.836	0.614	0.725	0.043	20.286	13.854	−10.397	−8.212	5.110	3.861	3.891	0.327	0.471	0.268	0.455

Table 5. Constant threshold-based stock classification with different $\mathbf{W}_{Loss}$ values and different features (all: all features; price: price-related features; trading volume: trading volume feature; financial status: company financial status-related features).

Tests.	True Negative Rate	True Positive Rate (Recall)	Average Correction Rate	Precision	Average of Maximum Rising Rate of Stocks Detected as Positive Samples and All (%)		Average of Maximum Decreasing Rate of Stocks Detected as Positive Samples and All (%)		Earning Rate of a Simulated System for Stocks Detected as Positive Samples and All (%)		Basis of All Stocks (%)	Sharpe ratio$_{T}$ of a Simulated System for Stocks Detected as Positive Samples and All		Sharpe Ratio$_{NT}$ of a Simulated System for Stocks Detected as Positive Samples and All	
					Detected	All	Detected	All	Detected	All		Detected	All	Detected	All
$\mathbf{W}_{Loss}$ (1:1), all	0.836	0.614	0.725	0.043	20.286	13.854	−10.397	−8.212	5.110	3.861	3.891	0.327	0.471	0.268	0.455
$\mathbf{W}_{Loss}$ (1:2), all	0.894	0.510	0.702	0.056	22.572	13.854	−11.447	−8.212	5.365	3.861	3.891	0.259	0.471	0.178	0.455
$\mathbf{W}_{Loss}$ (1:3), all	0.910	0.454	0.682	0.055	22.960	13.854	−11.594	−8.212	5.336	3.861	3.891	0.261	0.471	0.184	0.455
$\mathbf{W}_{Loss}$ (1:2), price	0.878	0.586	0.732	0.058	23.236	13.854	−11.447	−8.212	**6.037**	3.861	3.891	0.299	0.471	0.222	0.455
$\mathbf{W}_{Loss}$ (1:2), price and trading volume	0.881	0.578	0.730	0.057	22.876	13.854	−11.015	−8.212	5.924	3.861	3.891	0.317	0.471	0.239	0.455
$\mathbf{W}_{Loss}$ (1:2), price and trading volume and financial status	0.863	0.587	0.725	0.053	21.962	13.854	−10.916	−8.212	5.670	3.861	3.891	0.303	0.471	0.232	0.455

Table 6. Ranking-based stock classification with different $\mathbf{W}_{Loss}$ values and different features (all: all features; price: price-related features; trading volume: trading volume feature; financial status: company financial status-related features).

Tests.	True Negative Rate	True Positive Rate (Recall)	Average Correction Rate	Precision	Average of Maximum Rising Rate of Stocks Detected as Positive Samples and All (%)		Average of Maximum Decreasing Rate of Stocks Detected as Positive Samples and All (%)		Earning rate of a Simulated System for Stocks Detected as Positive Samples and All (%)		Basis of All Stocks (%)	Sharpe Ratio$_{NT}$ of a Simulated System for Stocks Detected as Positive Samples and All	
					Detected	All	Detected	All	Detected	All		Detected	All
$\mathbf{W}_{Loss}$ (1:1), all	0.871	0.186	0.528	0.154	19.848	13.854	−10.908	−8.212	4.886	3.891	3.891	0.241	0.455
$\mathbf{W}_{Loss}$ (1:2), all	0.982	0.039	0.510	0.206	26.667	13.854	−12.800	−8.212	7.656	3.891	3.891	0.152	0.455
$\mathbf{W}_{Loss}$ (1:3), all	0.985	0.028	0.506	0.161	24.347	13.854	−12.836	−8.212	5.615	3.891	3.891	0.021	0.455
$\mathbf{W}_{Loss}$ (1:2), price	0.992	0.028	0.510	0.146	22.372	13.854	−9.422	−8.212	4.852	3.891	3.891	0.098	0.455
$\mathbf{W}_{Loss}$ (1:2), price and trading volume	0.993	0.034	0.513	0.226	31.077	13.854	−11.089	−8.212	**10.875**	3.891	3.891	0.314	0.455
$\mathbf{W}_{Loss}$ (1:2), price and trading volume and financial status	0.964	0.098	0.531	0.210	25.855	13.854	−10.930	−8.212	8.853	3.891	3.891	0.460	0.455

Table 7. Constant threshold-based stock classification considering historical volatility using different $\mathbf{W}_{Loss}$ values and different features (all: all features; price: price-related features; trading volume: trading volume feature; financial status: company financial status-related features).

Tests.	True Negative Rate	True Positive Rate (Recall)	Average Correction Rate	Precision	Average of Maximum Rising Rate of Stocks Detected as Positive Samples and All (%)		Average of Maximum Decreasing Rate of Stocks Detected as Positive Samples and All (%)		Earning rate of a Simulated System for Stocks Detected as Positive Samples and All (%)		Basis of All Stocks (%)	Sharpe ratio$_T$ of a Simulated System for Stocks Detected as Positive Samples and All		Sharpe ratio$_{NT}$ of a Simulated System for Stocks Detected as Positive Samples and All	
					Detected	All	Detected	All	Detected	All		Detected	All	Detected	All
$\mathbf{W}_{Loss}$ (1:1), all	0.875	0.221	0.548	0.154	16.145	13.854	−7.369	−8.212	5.223	3.861	3.891	0.599	0.471	0.563	0.455
$\mathbf{W}_{Loss}$ (1:2), all	0.987	0.027	0.507	0.165	16.342	13.854	−7.838	−8.212	5.111	3.861	3.891	0.546	0.471	0.504	0.455
$\mathbf{W}_{Loss}$ (1:0.5), all	0.001	0.998	0.005	0.101	13.841	13.854	−8.208	−8.212	3.812	3.861	3.891	0.489	0.471	0.456	0.455
$\mathbf{W}_{Loss}$ (1:1), price	0.927	0.108	0.518	0.132	17.840	13.854	−10.681	−8.212	2.878	3.861	3.891	0.403	0.471	0.320	0.455
$\mathbf{W}_{Loss}$ (1:1), price and trading volume	0.881	0.226	0.553	0.180	17.810	13.854	−7.276	−8.212	**6.395**	3.861	3.891	0.641	0.471	0.594	0.455
$\mathbf{W}_{Loss}$ (1:1), price and trading volume and financial status	0.854	0.264	0.559	0.161	16.455	13.854	−7.387	−8.212	5.362	3.861	3.891	0.621	0.471	0.548	0.455

4.5. Results of Ranking-Based Stock Classification Considering Historical Volatility

Similar to the previous methods, this study also presents multiple evaluations for the ranking-based stock classification considering historical volatility. In the evaluation, the effect of different values of parameter $\mathbf{W}_{Loss}$ and input features was tested. Table 8 shows the results using different configurations. In this method, the top 10% of stocks were considered as positive samples. Table 8 shows a summary of the evaluations. There was no fixed threshold for stock trading; therefore, the simulated stock trading system kept the detected or all stocks until the end of next 12 weeks, then sold them. Thus, there was no Sharpe ratio with trading. The following conclusion can be obtained from the data in Table 8.

(1) Values of parameter $\mathbf{W}_{Loss}$ affect the earning rate.

(2) Price and trading volume-related features can provide the best performance compared to all features.

(3) The best earning rate happens when the stock classification uses $\mathbf{W}_{Loss}$ (1:1), price and trading volume-related features. The method of the ranking-based stock classification considering historical volatility has a 9.044% earning rate per 12 weeks in Japanese stock data from 2011 to 2018. A 5.153% (9.044–3.891%) earning per 12 weeks above basis is achieved in ranking-based stock classification considering historical volatility.

(4) The Sharpe ratio of the detected stock is higher than all stocks, which indicates this method can reduce the risk in stock classification. The value of Sharpe ratio of this method is very similar to its value in the method of the constant threshold-based stock classification considering historical volatility.

(5) By considering both earning rate and Sharpe ratio, the ranking-based stock classification method considering historical volatility is the most effective strategy among the proposed four strategies.

In addition, in this study, the proposed deep neural network-based stock performance prediction method was compared with two conventional methods: Logistic Regression-based classification and Support Vector Machine (SVM)-based classification. This comparison was performed for the ranking-based stock classification considering historical volatility. The comparison in Table 9 shows the proposed method has a higher earning rate and a lower risk than the conventional methods.

Table 8. Ranking-based stock classification considering historical volatility using different $\mathbf{W}_{Loss}$ values and different features (all: all features; price: price-related features; trading volume: trading volume feature; financial status: company financial status-related features).

Tests	True Negative Rate	True Positive Rate (Recall)	Average Correction Rate	Precision	Average of Maximum Rising Rate of Stocks Detected as Positive Samples and All (%)		Average of Maximum Decreasing Rate of Stocks Detected as Positive Samples and All (%)		Earning Rate of a Simulated System for Stocks Detected as Positive Samples and All (%)		Basis of All Stocks (%)	Sharpe ratio$_{NT}$ of a Simulated System for Stocks Detected as Positive Samples and All	
					Detected	All	Detected	All	Detected	All		Detected	All
$\mathbf{W}_{Loss}$ (1:1), all	0.918	0.103	0.511	0.119	17.213	13.854	−8.212	−8.212	5.639	3.891	3.891	0.522	0.455
$\mathbf{W}_{Loss}$ (1:2), all	0.998	0.003	0.500	0.096	13.060	13.854	−8.746	−8.212	2.516	3.891	3.891	0.141	0.455
$\mathbf{W}_{Loss}$ (1:0.5), all	0.004	0.997	0.500	0.100	13.829	13.854	−8.191	−8.212	3.903	3.891	3.891	0.458	0.455
$\mathbf{W}_{Loss}$ (1:1), price	0.916	0.108	0.512	0.096	17.696	13.854	−5.97	−8.212	8.448	3.891	3.891	0.706	0.455
$\mathbf{W}_{Loss}$ (1:1), price and trading volume	0.975	0.041	0.508	0.154	23.365	13.854	−8.226	−8.212	**9.044**	3.891	3.891	0.665	0.455
$\mathbf{W}_{Loss}$ (1:1), price and trading volume and financial status	0.936	0.102	0.519	0.134	18.846	13.854	−8.176	−8.212	6.480	3.891	3.891	0.549	0.455

Table 9. Comparison for ranking-based stock classification considering historical volatility.

Tests.	True Negative Rate	True Positive Rate (Recall)	Average Correction Rate	Precision	Average of Maximum Rising Rate of Stocks Detected as Positive Samples and All (%)		Average of Maximum Decreasing Rate of Stocks Detected as Positive Samples and All (%)		Earning Rate of a Simulated System for Stocks Detected as Positive Samples and All (%)		Basis of all stocks (%)	Sharpe ratio$_{NT}$ of a Simulated System for Stocks Detected as Positive Samples and All	
					Detected	All	Detected	All	Detected	All		Detected	All
Proposed method	0.975	0.041	0.508	0.154	23.365	13.854	−8.226	−8.212	**9.044**	3.891	3.891	0.665	0.455
Logistic Regression	0.046	0.937	0.491	0.098	13.783	13.854	−8.166	−8.212	3.857	3.891	3.891	0.457	0.455
SVM	0.117	0.833	0.475	0.094	14.088	13.854	−8.365	−8.212	3.928	3.891	3.891	0.449	0.455

5. Conclusions

This study presents four strategies for stock classification and performs feature selection to achieve a higher earning rate and lower risk in stock classification. The following points are concluded based on the evaluations and analysis:

(1) Using the ranking method can improve the earning rate above basis.
(2) Using historical volatility information for training can increase the Sharpe ratio and reduce the risk in stock classification.
(3) Using price and trading volume-related features has a better performance than using full features in stock classification.
(4) By considering both earning rate and risk, the ranking-based stock classification method considering historical volatility is the most effective strategy among the proposed four methods. About 5.2% earning rate per 12 weeks over than the basis of all stocks is achieved in the repeatability test. The selected stocks have lower risky than basis.
(5) The proposed deep neural network-based stock performance prediction method has better performance than the conventional methods.

The proposed method can have about 36% (9.044% per 12 weeks) earning rate per year in the Japanese stock market. However, excellent human stock traders have achieved triple-digit returns per year [6]. There is still a gap between the performance of the developed intelligent trading system and the achievements of human stock traders. This paper proposed the use of the classification network to classify the stocks into two categories: buying or not. In the future, a regression network will be developed to predict the exact value of the future price. In this way, the developed trading system is expected to obtain higher earnings.

Author Contributions: Conceptualization, S.K. and S.N.; methodology, S.K., S.N., T.S., Y.K. and Y.G.; data acquisition, T.S. and Y.G.; software, Y.G.; experiment, Y.G.; writing, Y.G.; project administration, S.K. and S.N.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Minervini, M. Trade like a Stock Market Wizard: How to Achieve Super Performance in Stocks in Any Market. McGraw-Hill: New York, NY, USA, 2013.
2. Schwager, J.D. *Market Wizards: Interviews with Top Traders*; Wiley: Hoboken, NJ, USA, 2012.
3. Schwager, J.D. *Stock Market Wizards: Interviews with America's Top Stock Traders*; Harper-Business: New York, NY, USA, 2003.
4. William, J.O. *How to Make Money in Stocks: A Winning System in Good Times and Bad*, 4th ed.; McGraw-Hill: New York, NY, USA, 2009.
5. Book Review of the Successful Investor by William J. O'Neil. Available online: https://whatheheckaboom.wordpress.com/2012/04/23/book-review-of-the-successful-investor-by-william-j-oneil/ (accessed on 1 September 2020).
6. Book description of Trade like a Stock Market Wizard: How to Achieve Super Performance in Stocks in Any Market. Available online: https://www.mhebooklibrary.com/doi/book/10.1036/9780071807234 (accessed on 1 September 2020).
7. Martinez, L.C.; Da Hora, D.N.; Palotti, J.R.D.M.; Meira, W.; Pappa, G.L. From an Artificial Neural Network to a Stock Market Day-trading System: A Case Study on the BM&F BOVESPA. In Proceedings of the 2009 International Joint Conference on Neural Networks, Atlanta, GA, USA, 14–19 June 2009.
8. Dhar, S.; Mukherjee, T.; Ghoshal, A.K. Performance Evaluation of Neural Network Approach in Financial Prediction: Evidence from Indian Market. In Proceedings of the 2010 International Conference on Communication and Computational Intelligence, Erode, India, 27–29 December 2010.

9. Oliveira, A.L.; Meira, S.R. Detecting Novelties in Time Series Through Neural Networks Forecasting with Robust Confidence Intervals. *Neurocomputing* **2006**, *70*, 79–92. [CrossRef]
10. Chen, A.S.; Leung, M.T.; Daouk, H. Application of Neural Networks to an Emerging Financial Market: Forecasting and Trading the Taiwan Stock Index. *Comput. Oper. Res.* **2003**, *30*, 901–923. [CrossRef]
11. Guresen, E.; Kayakutlu, G.; Daim, T.U. Using Artificial Neural Network Models in Stock Market Index Prediction. *Expert Syst. Appl.* **2011**, *38*, 10389–10397. [CrossRef]
12. Sezer, O.B.; Ozbayoglu, A.M.; Dogdu, E. An Artificial Neural Network-based Stock Trading System using Technical Analysis and Big Data Framework. In Proceedings of the SouthEast Conference, Kennesaw, GA, USA, 13–15 April 2017.
13. Prasaddas, S.; Padhy, S. Support Vector Machines for Prediction of Futures Prices in Indian Stock Market. *Int. J. Comput. Appl.* **2012**, *41*, 22–26. [CrossRef]
14. Bao, Y.; Yang, Y.; Xiong, T.; Zhang, J. A Comparative Study of Multi-step-ahead Prediction for Crude Oil Price with Support Vector Regression. In Proceedings of the IEEE Fourth International Joint Conference on Computational Sciences and Optimization, Yunnan, China, 15–19 April 2011.
15. Kara, Y.; Acar Boyacioglu, M.; Baykan, O.K. Predicting Direction of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines: The Sample of the Istanbul Stock Exchange. *Expert Syst. Appl.* **2011**, *38*, 5311–5319. [CrossRef]
16. Aguilar-Rivera, R.; Valenzuela-Rendón, M.; Rodriguez-Ortiz, J. Genetic Algorithms and Darwinian Approaches in Financial Applications: A Survey. *Expert Syst. Appl.* **2015**, *42*, 7684–7697. [CrossRef]
17. Nayak, R.K.; Mishra, D.; Rath, A.K. A Naive SVM-KNN based Stock Market Trend Reversal Analysis for Indian Benchmark Indices. *Appl. Soft Comput.* **2015**, *35*, 670–680. [CrossRef]
18. Cai, X.; Xu, T.; Yi, J.; Huang, J.; Rajasekaran, S. DTWNet: A Dynamic Time Warping Network. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 11640–11650.
19. Ramos-Requena, J.P.; Trinidad-Segovia, J.E.; Sánchez-Granero, M.Á. An Alternative Approach to Measure Co-Movement between Two Time Series. *Mathematics* **2020**, *8*, 261. [CrossRef]
20. Krollner, B.; Vanstone, B.; Finnie, G. Financial Time Series Forecasting with Machine Learning Techniques: A Survey. In Proceedings of the 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, 28–30 April 2010.
21. Cavalcante, R.C.; Brasileiro, R.C.; Souza, V.L.; Nobrega, J.P.; Oliveira, A.L. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Syst. Appl.* **2016**, *55*, 194–211. [CrossRef]
22. Nakagawa, K.; Uchida, T.; Aoshima, T. Deep factor model Explaining Deep Learning Decisions for Forecasting Stock Returns with Layer-Wise Relevance Propagation. In Proceedings of the ECML PKDD 2018 Workshops, Dublin, Ireland, 10–14 September 2018.
23. Abe, M.; Nakayama, H. Deep Learning for Forecasting Stock Returns in the Cross-section. In Proceedings of the 2018 Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, 3–6 June 2018.
24. Rout, A.K.; Dash, P.K.; Dash, R.; Bisoi, R. Forecasting Financial Time Series using a Low Complexity Recurrent Neural Network and Evolutionary Learning Approach. *J. King Saud Univ. -Comput. Inf. Sci.* **2017**, *29*, 536–552. [CrossRef]
25. Dos Santos Pinheiro, L.; Dras, M. Stock Market Prediction with Deep Learning: A Character-based Neural Language Model for Event-based Trading. In Proceedings of the Australasian Language Technology Association Workshop 2017, Brisbane, Australia, 6–8 December 2017.
26. Li, J.; Bu, H.; Wu, J. Sentiment-aware Stock Market Prediction: A Deep Learning Method. In Proceedings of the IEEE 2017 International Conference on Service Systems and Service Management, Dalian, China, 16–18 June 2017.
27. Nelson, D.M.; Pereira, A.C.; De Oliveira, R.A. Stock Market's Price Movement Prediction with LSTM Neural Networks. In Proceedings of the 2017 International Joint Conference on Neural Networks, Anchorage, AK, USA, 14–19 May 2017.
28. Bao, W.; Yue, J.; Rao, Y. A Deep Learning Framework for Financial Time Series using Stacked Autoencoders and Long-short Term Memory. *PLoS ONE* **2017**, *12*, e0180944. [CrossRef]
29. Fischer, T.; Krauss, C. Deep Learning with Long Short-term Memory Networks for Financial Market Predictions. *Eur. J. Oper. Res.* **2018**, *270*, 654–669. [CrossRef]

30. Nakagawa, K.; Ito, T.; Abe, M.; Izumi, K. Deep Recurrent Factor Model: Interpretable Non-Linear and Time-Varying Multi-Factor Model. In Proceedings of the AAAI 2019 Workshop on Network Interpretability for Deep Learning, Honolulu, HI, USA, 27–28 January 2019; 2019.
31. Sezer, O.B.; Ozbayoglu, A.M. Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach. *Appl. Soft Comput.* **2018**, *70*, 525–538. [CrossRef]
32. Zhou, X.; Pan, Z.; Hu, G.; Tang, S.; Zhao, C. Stock Market Prediction on High-frequency Data using Generative Adversarial Nets. *Math. Probl. Eng.* **2018**, *2018*, 525–538. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Hybrid Forecasting Models Based on the Neural Networks for the Volatility of Bitcoin

Monghwan Seo [1] and Geonwoo Kim [2,*]

[1] Department of Mathematics, Yonsei University, 50 Yonsei-ro Seodaemun-gu, Seoul 03722, Korea; smh3261@naver.com

[2] School of Liberal Arts, Seoul National University of Science and Technology, Seoul 01811, Korea

* Correspondence: geonwoo@seoultech.ac.kr

Received: 10 June 2020; Accepted: 9 July 2020; Published: 10 July 2020

Abstract: In this paper, we study the volatility forecasts in the Bitcoin market, which has become popular in the global market in recent years. Since the volatility forecasts help trading decisions of traders who want a profit, the volatility forecasting is an important task in the market. For the improvement of the forecasting accuracy of Bitcoin's volatility, we develop the hybrid forecasting models combining the GARCH family models with the machine learning (ML) approach. Specifically, we adopt Artificial Neural Network (ANN) and Higher Order Neural Network (HONN) for the ML approach and construct the hybrid models using the outputs of the GARCH models and several relevant variables as input variables. We carry out many experiments based on the proposed models and compare the forecasting accuracy of the models. In addition, we provide the Model Confidence Set (MCS) test to find statistically the best model. The results show that the hybrid models based on HONN provide more accurate forecasts than the other models.

Keywords: Bitcoin; artificial neural network; higher order neural network; volatility forecasting; hybrid models

1. Introduction and Review of Models

1.1. Introduction

Online transactions over the Internet have depended on trusted financial institutions, which are central players for safe transactions. Nakamoto [1] proposed Bitcoin as a digital currency to provide an easy method to perform online transactions. Bitcoin is a peer-to-peer crypocurrency system, where Bitcoin transactions occur with no central players. All Bitcoin transactions are verified by the nodes of the peer-to-peer networks and added to the blockchain as the Bitcoin ledger. The information of all historical transactions and all Bitcoin clients is stored in the blackchain. That is, Bitcoin transactions are recorded in the blockchain. The value of Bitcoin is not based on the economic condition in any country and depends on only the supply and demand of the network. Thus, Bitcoin has been utilized widely as a digital currency that can be exchanged for real products or services based on the Bitcoin market value. In fact, there are various digital currencies such as Ethereum, Ripple, Stellar, etc. However, we focus only on Bitcoin because the Bitcoin market capitalization is about 50% of the total estimated digital currency capitalization at present.

As the Bitcoin market has grown over the years, there have been many studies to analyze the Bitcoin market in recent years. Urquhart [2] studied the efficiency of Bitcoin market. In an efficient market, due to the random nature of unpredictable events, variations are random. To find the inefficiency, Urquhart employed a battery of highly powerful tests for randomness and found evidence of inefficiency. The high-frequency multifractal properties of Bitcoin were examined in [3]. Gajardo et al. [4] analyzed the asymmetric multifractal cross-correlations among stock market indices, commodities and Bitcoin. Yonghong et al. [5] also investigated the time-varying long-term memory in the Bitcoin market. Dyhrberg [6,7] showed that Bitcoin has a clear role in the market for portfolio management. Some researchers studied Bitcoin as an investment vehicle [8–10]. They found out that Bitcoin investment has characteristic features such as high average return and volatility. Although the volatilities of various financial indices have an important impact on the Bitcoin market, the most important factor that affects the high volatility of Bitcoin is the speculative behavior of users. In addition, there was a study on economic analyses of Bitcoin as a currency [11]. According to Iwamura et al. [11] and Yermack [12], Bitcoin may not be suitable as currency since Bitcoin has high volatility. Baur et al. [13] also showed that Bitcoin is used as a speculative investment due to high volatility and large returns. In practice, since the Bitcoin market has high volatility, the study on the volatility of Bitcoin has been very important. We focus on the volatility of Bitcoin in this paper. Specifically, we study the accurate methods for forecasting of Bitcoin volatility.

Many researchers have investigated the analysis and prediction of Bitcoin volatility recently. Baur and Dimpfl [14] analyzed asymmetric volatility effects for Bitcoin. Other studies attempted to show that Bitcoin volatility has some properties such as chaos, randomness, multi-fractality and long-range memory [15,16]. Additionally, there have been many studies on the forecasting of Bitcoin volatility. Balcilar et al. [17] studied the prediction of Bitcoin volatility with a quantile test based on the trading volume. Katsiampa [18] investigated several GARCH family models to find the best model for Bitcoin volatility and found that the AR-CGARCH is the optimal model. Chu et al. [19] provided the best fitting models based on GARCH models for volatilities of cryptocurrencies including Bitcoin. They fit 12 GARCH models to each cryptocurrency and found that IGARCH (1,1) model provides a good fit. Conrad et al. [20] used the GARCH-MIDAS model to improve the prediction of long-term Bitcoin volatility. However, GARCH models have limitations that are hard to capture complex fluctuation and nonlinear correlation of time series data. In order to overcome these limitations, many researchers have proposed the non-parametric forecasting methods based on machine learning approaches such as ANN for better forecasting of Bitcoin volatility [21–23].

Over the past few years, there have been various hybrid models based on ANN to improve the forecasting ability of the time series data. In particular, the hybrid models based on ANN and GARCH models have been proposed to improve forecast accuracy for the time-series data such as market indices, exchange rate, stock volatility, gold price, oil price and metal, etc. [24–30]. These results have shown that the hybrid models have an advantage compared to ANN models. The so-called ANN-GARCH models are the hybrid models that incorporate the GARCH forecasts as the explanatory variables to the ANN models and have been developed consistently by many researchers. For instance, Hajizadeh et al. [31] proposed two ANN-GARCH models to improve the forecasting performance of the S&P 500 index volatility. They used various input variables including financial indicators and the simulated volatility by GARCH models, and the proposed hybrid model with EGARCH model show better accuracy than the traditional GARCH models and ANN models. Kristjanpoller et al. [32] provided the methodology and the application for the volatility forecast of three Latin American stock indexes using a hybrid ANN-GARCH model. Lahmiri and Boukadoum [33] presented an ensemble system based on a hybrid EGARCH-ANN model which is trained with a different distributional assumption. In addition, Seo et al. [34] constructed the hybrid ANN-GARCH model with Google domestic trend and various activation functions for better forecasting accuracy of S&P 500 index volatility. In this paper, we also employ the ANN-GARCH models for accurate forecasting of the realized volatility of Bitcoin. Specifically, we develop ANN-GARCH models with HONN and Google trends (GT) data and compare the proposed models to find the best fitting model for Bitcoin volatility.

The contribution of this work is to find the optimal hybrid model for forecasting Bitcoin's volatility. To present our result, this paper is structured as follows. In the next subsection, we review the models used in this paper. In Section 2, we describe the data used for the proposed hybrid models. In Section 3, we construct efficient hybrid models and provide the results of the experiments by the proposed models. In Section 4, we present the concluding remarks.

1.2. Review of Models

In this section, we introduce GARCH family models used to construct our hybrid models. More specifically, we review the GARCH model, EGARCH model and GJR-GARCH model. The forecasts by GARCH family models are used as the explanatory variables to ANN. We also review ANN model and HONN model with various activation functions used in this paper.

1.2.1. GARCH Model

The ARCH model proposed by Engle [35] was the first model with the conditional distribution to describe the fat tail characteristics or the volatility clustering properties of time series. However, the ARCH model has computational problems when a large number of parameters are needed for a high order model. To solve these problems, Bollerslev [36] proposed the GARCH model, which is one of the most popular models for forecasting the volatility of time series. Since the GARCH models include the conditional variance terms as well as the squared residual terms, the models can predict the volatility well by using a sum of weighted products of the predicted variance from the past.

The GARCH (p, q) model is defined as the follows.

$$y_t^2 = w + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i y_{t-i}^2, \tag{1}$$

where $\varepsilon_t = y_t Z_t$, $\{Z_t\}$ is a sequence of independent and identically distributed random variables with zero mean and unit variance, $\{\varepsilon_t\}$ is a sequence of the error terms, the positive parameters α_i and β_i satisfy the condition $\sum_{i=1}^{q} \alpha_i + \sum_{i=1}^{p} \beta_i < 1$ for the stability of the GARCH model. This condition ensures that the conditional variance y_t has nonnegative values and finite expected value. Here, w, α_i and β_i are the estimated parameters by using maximum likelihood estimation.

1.2.2. EGARCH Model

The exponential GARCH (EGARCH) model proposed by Nelson [37] allows negative parameters unlike the GARCH model. That is, the parameters of the model have no restrictions to ensure the non-negativity of the volatility. This model can describe the volatility leverage effect which reflects the asymmetric impacts and captures asymmetric behavior of the time series.

The EGARCH (p, q) model is defined as follows.

$$\log y_t^2 = w + \sum_{i=1}^{q} \alpha_i \left[\frac{|\varepsilon_{t-i}|}{y_{t-i}} - \sqrt{\frac{2}{\pi}} + \gamma \frac{\varepsilon_{t-i}}{y_{t-i}} \right] + \sum_{i=1}^{p} \beta_i \log y_{t-i}^2, \tag{2}$$

where α_i with no restrictions captures the volatility clustering effect, β_i measures the persistence in conditional volatility irrespective of the events in the market and γ measures the asymmetric leverage coefficient to describe the leverage effect of volatility. α_i, β_i and γ are parameters to be estimated.

1.2.3. GJR-GARCH Model

The GJR-GARCH model proposed by Glosten et al. [38] is one of nonlinear GARCH family models to allow for asymmetry effects by integrating a dichotomous variable into the GARCH model. This model allows the larger impact of negative shocks to have a more distinct impact on volatility than a positive impact. The model also presented improved forecasting ability [39].

The conditional variance of GJR-GARCH (p, q) model is defined as follows.

$$y_t^2 = w + \sum_{i=1}^{q} \left[\alpha_i + \gamma_i \mathbf{1}_{\{\varepsilon_{t-i} < 0\}} \right] \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i y_{t-i}^2, \tag{3}$$

where

$$\mathbf{1}_{\{\cdot\}} = \begin{cases} 1, & \varepsilon_{t-i} < 0, \\ 0, & \varepsilon_{t-i} \geq 0, \end{cases}$$

and

$$w \geq 0, p \geq 0, q \geq 0, \alpha_i \geq 0, \beta_i \geq 0, \alpha_i + \gamma_i \geq 0 \text{ and } \sum_{i=1}^{p} \alpha_i + \sum_{i=1}^{q} \beta_i + \frac{1}{2} \sum_{i=1}^{q} \gamma_i < 1.$$

where α_i and β_i are similar to the coefficients in the EGARCH model, and γ_i means the asymmetric leverage coefficient. The parameters w, α_i, β_i and γ_i are estimated by the maximum likelihood approach.

1.2.4. Artificial Neural Network (ANN)

ANN is one of the nonparametric nonlinear models which are used widely to overcome the limitations of the linear models in machine learning. ANN is constructed appropriately based on the characteristics extracted from the real data and has no hypothesis about the underlying model. ANN also has at least three layers (input layer, hidden layer, output layer). ANN with single hidden layer used for forecasting is illustrated in Figure 1.

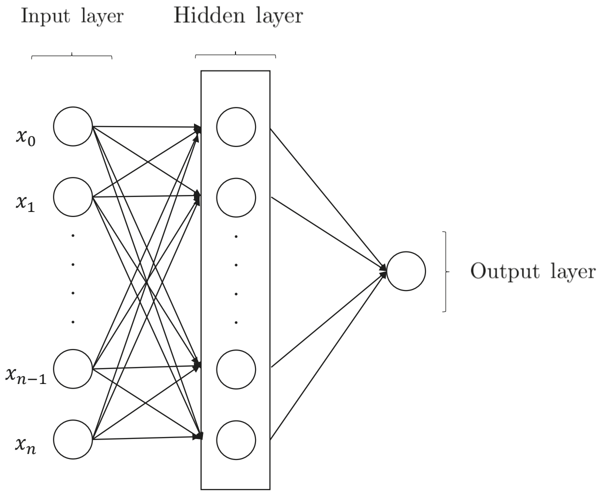


Figure 1. The structure of Artificial Neural Network (ANN).

The output result from input layer and hidden layer is generally as follows.

$$\text{output} = f\left(\sum_{i=0}^{n} x_i w_i\right), \tag{4}$$

where x_i and w_i represent the set of input data from node i and the weight associated with the connection to the node i, and f is one of the activation functions. The activation functions used in this paper are presented in Table 1. The sigmoid function shows high sensitivity to small changes in input variables. This property provides a good classifier. The hyperbolic tangent function (Tanh) has an advantage over the sigmoid function. Since the derivative of the function is steeper, it will have faster learning and grading. In addition, it is well known that the Rectified Linear Unit (ReLU) is a good estimator and show very efficient calculation when all neurons are activated in the same manner. Exponential Linear Unit (ELU) provides fast learning because ELU shrinks the difference between the unit natural gradient and the normal gradient.

Table 1. Activation functions used in this paper.

Name	Activation Function
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Hyperbolic Tangent (Tanh)	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Rectified Linear Unit (ReLU)	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)	$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

The main work of ANN is to find the optimal weights for better performance using the activation functions. We use the back-propagation method to obtain the weights. We also carry out many experiments with four activation functions to find the best forecasting model.

1.2.5. Higher Order Neural Network (HONN)

HONN proposed by Giles and Maxwell [40] has been widely used to simulate the higher-order nonlinear inputs and to provide some basis for the simulations as 'open box' [41]. Because first-order networks do not take advantage of meaningful relationships between the input variables, the networks need a lot of training passes with a large training set. To improve this disadvantage, HONN has been developed. In general, with the selection of good input variables, it is known that HONN provides better forecasting performance than the classic ANN.

In Equation (4), the independent variable is presented as the linear combination. Specifically, the variable is expressed by multiplying each input variable (x_i) by a weight (w_i) and adding the results. We can easily make out the higher-order terms of the inputs from the first-order terms. Here, we consider the second order HONN to improve the volatility forecasting. Let us define the input vector $\vec{x}$ and the weight vector $\vec{w}$ by

$$\vec{x} = [x_0, x_1, \cdots, x_n] \text{ and } \vec{w} = [w_0, w_1, \cdots, w_n],$$

respectively. Then the input vector $\vec{x}_h$ and the weight vector $\vec{w}_h$ in HONN are given by

$$\vec{x}_h = [x_0, x_1, \cdots, x_n, x_0^2, x_0x_1, x_0x_2, \cdots, x_{n-1}x_n, x_n^2] \text{ and } \vec{w}_h = [w_0, w_1, \cdots, w_n, w_{00}, w_{01}, w_{02}, \cdots, w_{n-1n}, w_{nn}],$$

respectively. From these vectors, the output with the activation functions f can be calculated as follows.

$$\text{output} = f\left(\vec{w}_h \cdot \vec{x}_h\right) = f\left(\sum_{i=0}^{n} w_i x_i + \sum_{i=0}^{n} \sum_{j=i}^{n} w_{ij} x_i x_j\right). \tag{5}$$

The structure of a second-order HONN used in this paper is illustrated in Figure 2. We construct the hybrid models based on this second-order HONN for the accurate forecasting.

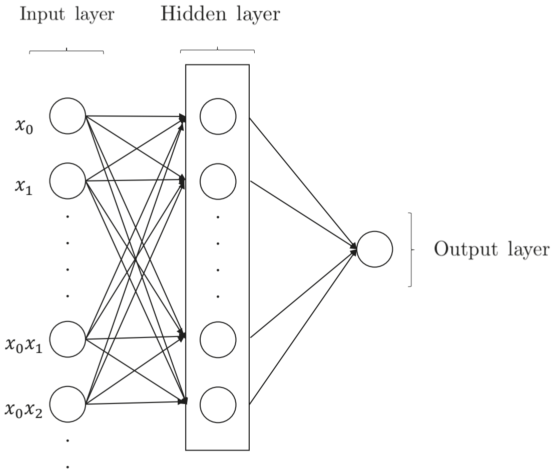


Figure 2. The structure of Higher Order Neural Network (HONN).

2. Material and Methods

The time series data analyzed in this paper were the daily historical prices of Bitcoin over the period between 1 January 2012 and 30 November 2019. The data were downloaded from the website (https://bitcoincharts.com/). To define the volatility of Bitcoin price, the closing prices p_t at time t are transformed into log return $r_t = \log p_t - \log p_{t-1}$. The realized volatility of Bitcoin was computed as the variance of r_t, and the realized volatilities in a 5-day window as weekly volatilities are used to analyze the volatility of Bitcoin in this paper. Then, the realized volatility (RV_t) of Bitcoin at time t is computed as

$$RV_t = \frac{1}{5}\sum_{i=t+1}^{t+5}(r_i - \bar{r}_t)^2,$$

where $\bar{r}_t$ is mean of r_t during 5 days after time t.

In order to improve the accuracy of the volatility forecast, the selection of the input data which influence on the volatility of Bitcoin is very important. In this paper, we consider the GT data and VIX data as the explanatory variables. GT is the data that presents the popularity of search queries related to various sectors in Google. In fact, GT data has been used as explanatory variables in the ANN to forecast of the financial time series by many researchers [34,42–44]. We used 'Bitcoin' GT data as the input variable, which is a good measure to describe the Bitcoin market [45]. VIX index introduced the Chicago Board Options Exchange (CBOE) in 2004 extrapolates the future volatility from the liquid options written on the S&P 500 and is calculated as the square root of the risk-neutral expectation of the 30 days variance of the S&P 500 return which is estimated by the forward option price expiring in 30 days. From the previous works [46,47], we can find the significant relationship between the VIX index and Bitcoin. Thus, we choose the VIX index as the input data to the ANN-based on the researches. Specifically, 5-days moving averages of VIX index and GT data are used as the input data. In Figure 3, the time series of log return r_t of Bitcoin price are displayed. Figures 4 and 5 illustrate the realized volatility of bitcoin price and VIX index, respectively.

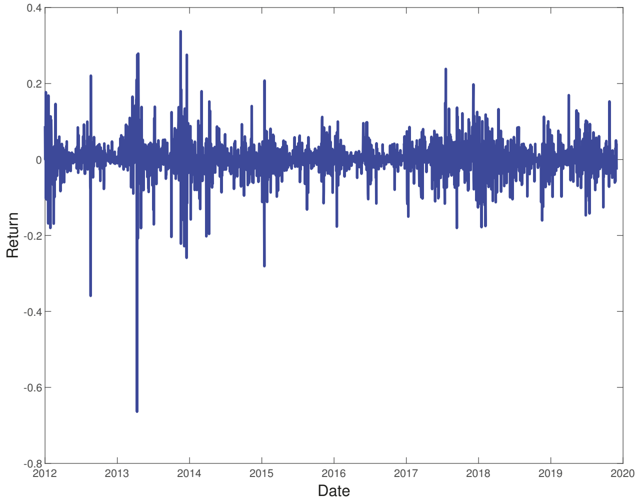


Figure 3. Log return r_t of Bitcoin price from 1 January 2012 to 30 November 2019.

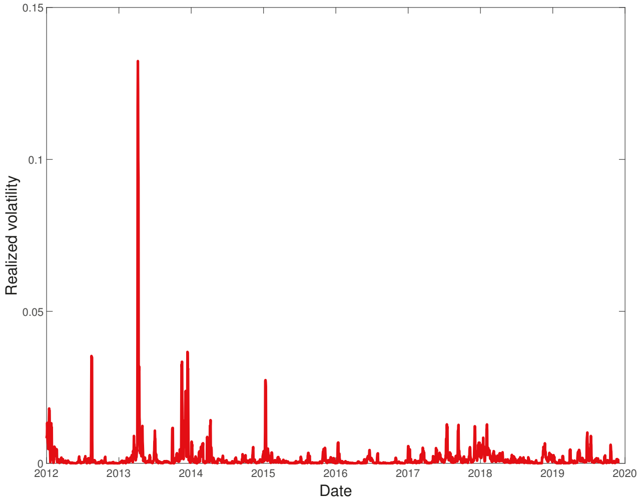


Figure 4. Realized volatility RV_t of r_t afrom 1 January 2012 to 30 November 2019.

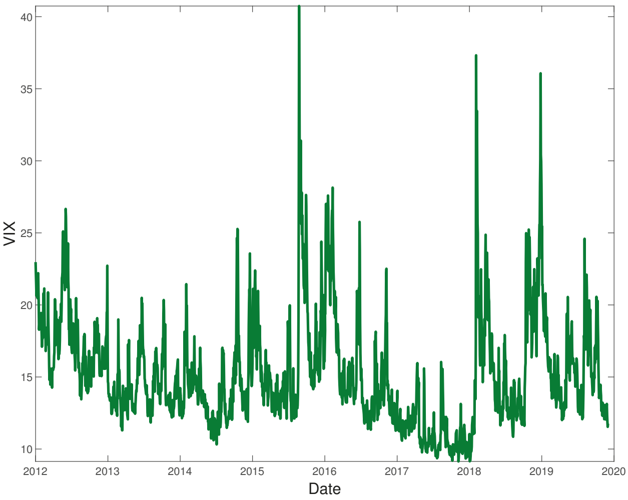


Figure 5. VIX index from 1 January 2012 to 30 November 2019.

In order to construct a more accurate model for forecasting of Bitcoin volatility, we use the 1-day lagged weekly volatility (LV_t) as the endogenous variable and the outputs of GARCH family models as the exogenous variables. In other words, LV_t and GARCH family outputs are used as the input variables to improve the forecasting ability of the hybrid model. Here, the outputs of the GARCH models introduced in the previous section are used, and LV_t is calculated by

$$LV_t = \frac{1}{5}\sum_{i=t-1}^{t-5}(r_i - \bar{r}_t)^2. \tag{6}$$

Note that days in windows of LV_t have no intersection with 5 days in windows of RV_t. LV_t is displayed in Figure 6. In this study, 80% of the data set (in-sample: 2012.01.01–2018.04.30) are used for training, and 20% (out-of-sample: 2018.05.01–2019.11.30) of the data set are used for testing. All experiments are implemented using Python 3. Additionally, we utilize three measures to compare the performance of the proposed models. These measures are the mean absolute error (MAE), the root mean square error (RMSE) and the mean absolute percentage error (MAPE) and as follows.

$$\text{MAE} = \frac{1}{n}\sum_t |\hat{\sigma}_t - RV_t|,$$

$$\text{RMSE} = \left(\frac{1}{n}\sum_t (\hat{\sigma}_t - RV_t)^2\right)^{1/2},$$

$$\text{MAPE} = \frac{1}{n}\sum_t \left|\frac{\hat{\sigma}_t - RV_t}{RV_t}\right|,$$

where $\hat{\sigma}_t$ is the predicted volatility of Bitcoin and n is the number of the predicted data. Obviously, the lower values of the measures, the better accuracy of the model. For more details, see [48].

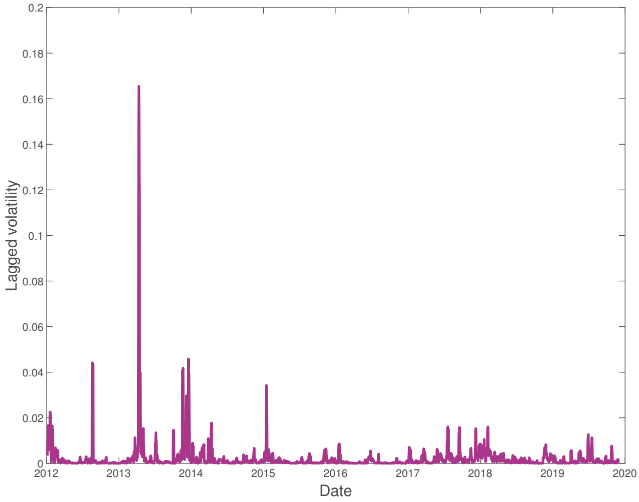


Figure 6. Lagged volatility LV_t of r_t from 1 January 2012 to 30 November 2019.

3. Hybrid Models and Results

In this paper, we propose several hybrid models based on GARCH family models, ANN and HONN to find a more accurate model for forecasting of Bitcoin volatility. Specifically, the hybrid models are constructed with the ANN by using the selected GARCH models and the selected explanatory variables. The models are implemented by the ANN with a single hidden layer and various neurons using the back-propagation method and classified according to whether including the explanatory variables or not. The proposed models are used for 1-day ahead forecast of weekly realized volatility, and then the best model is determined by comparing the results.

We compare the proposed models to find the best volatility forecasting model in the bitcoin market. We first forecast the volatility of Bitcoin price using the classic GARCH family models. Concretely, we use GARCH, EGARCH and GJR-GARCH model among the GARCH family models and the (p, q) parameters ranging from (1,1) to (3,3). In order to find the optimal GARCH model for the hybrid model, we provide AIC and BIC values in Table 2 and three measures to compare the performances of the models for forecasting volatilities in Table 3. According to the results in Table 2 and AIC and BIC criteria, EGARCH(3,3) model is the best model. On the other hand, according to the results in Table 3, we can see that the GJR-GARCH(1,1) model performs the best among the introduced GARCH family models.

Table 2. GARCH models.

Models	(p, q)	AIC	BIC
GARCH	(1,1)	−7593.56	−7570.83
GARCH	(2,2)	−7633.38	−7599.30
GARCH	(3,3)	−7630.73	−7585.28
GJR-GARCH	(1,1)	−7589.75	−7561.34
GJR-GARCH	(2,2)	−7577.91	−7538.14
GJR-GARCH	(3,3)	−7558.42	−7507.29
EGARCH	(1,1)	−7646.91	−7618.51
EGARCH	(2,2)	−7665.96	−7626.19
EGARCH	(3,3)	−7687.68	−7636.55

Table 3. GARCH models performance.

Models	(p, q)	MAE	RMSE	MAPE
GARCH	(1,1)	0.01820086	0.022997082	60.71728437
GARCH	(2,2)	0.018615039	0.023244302	62.97792289
GARCH	(3,3)	0.031275112	0.274933282	104.4275052
GJR-GARCH	(1,1)	0.018100989	0.022782066	59.57816469
GJR-GARCH	(2,2)	0.018273329	0.022976453	61.3172729
GJR-GARCH	(3,3)	0.018353907	0.023128309	61.44172661
EGARCH	(1,1)	0.021923047	0.025916869	80.21691954
EGARCH	(2,2)	0.021758949	0.025850653	79.23566863
EGARCH	(3,3)	0.022439612	0.026596278	81.70952727

Other models except for the classic GARCH models are based upon the ANN approach or the HONN approach. In other words, the models are constructed by using the selected input variables to ANN or HONN. Similar to [31,34], we propose the ANN-GARCH models for the forecasting of the Bitcoin volatility using the outputs of the GARCH family models. Specifically, we define the GT-GARCH model and GT-VIX-GARCH model according to the input variables. The input variables of the models are in Table 4. In order to find the optimal number of nodes in the hidden layer and the activation function for the models, we carry out the experiments using the Adam optimizer method [49] to update the network weights. The results are indicated with four activation functions in Tables 5 and 6. As shown in Tables 5 and 6, two measures (MAE, RMSE) show that the GT-GARCH model is better than the GT-VIX-GARCH model, and one measure (MAPE) shows a different result. From these results, we can not find a significant performance difference between the GT-VIX-GARCH model and the GT-GARCH model. That is, we conclude that two models may have a similar predictive ability. To improve the accuracy of the model, we adopt the HONN approach. Specifically, we propose three types of hybrid models (GT-H model, GT-VIX-H model, GT-VIX-GARCH-H model) based on the HONN.

Tables 7–9 are presented the results of the models based on the HONN. To examine well the proposed models based on the HONN, we present a summary of the input variables of each model in Table 10. In Table 10, 'LV_t' is in Equation (6), 'GT' means Google trends data, 'VIX' means VIX index data, 'GJR-GARCH(1,1)' means forecast by GJR-GARCH(1,1) and 'EGARCH(3,3)' means forecast by EGARCH(3,3). Tables 7 and 8 present the results of the HONN model without the outputs of GARCH models as shown in Table 10. We can see that MAE and MAPE in Tables 7 and 8 increase in all cases as compared to the values in Tables 5 and 6. That is, GT-H model and GT-VIX-H model do not show better performance compared to the models based on the ANN. To improve the model, we adopt the HONN model with the outputs of GARCH family models. Among the introduced GARCH models, we chose GJR-GARCH(1,1) and EGARCH(3,3) from the results in Tables 2 and 3. By using the outputs of GJR-GARCH(1,1) and EGARCH(3,3) as input variables in the HONN, we finally construct and propose a new type of hybrid model (GT-VIX-GARCH-H model) for better forecasting of Bitcoin volatility.

Table 4. Input variables of models.

Models	Selected Input Variables
GT-GARCH model	{GARCH(1,1), GARCH(2,2), GARCH(3,3), GJR-GARCH(1,1), GJR-GARCH(2,2), GJR-GARCH(3,3), EGARCH(1,1), EGARCH(2,2), EGARCH(3,3), GT, LV_t }
GT-VIX-GARCH model	{GARCH(1,1), GARCH(2,2), GARCH(3,3), GJR-GARCH(1,1), GJR-GARCH(2,2), GJR-GARCH(3,3), EGARCH(1,1), EGARCH(2,2), EGARCH(3,3), GT, LV_t, VIX }

Table 5. GT-GARCH model performance.

Model	Activation Function	Nodes	MAE	RMSE	MAPE
GT-GARCH	Relu	10	0.016455061	0.0228628	44.03939302
		20	0.016455761	0.02286762	44.01794052
		30	0.016456899	0.022870071	44.01377941
		40	0.016457765	0.02287134	44.01490369
		50	0.016456698	0.022868367	44.01481934
	Tanh	10	0.01645589	0.022866914	44.02158481
		20	0.016456046	0.022867024	44.02205182
		30	0.016457015	0.022869247	44.01766755
		40	0.016457969	0.022870135	44.0273439
		50	0.016457073	0.022869712	44.01422672
	Elu	10	0.016456301	0.022867778	44.01894413
		20	0.016451684	0.02284573	44.10871
		30	0.016456961	0.022866742	44.03292253
		40	0.016455294	0.022864722	44.02362704
		50	0.016456115	0.022866339	44.0296898
	Sigmoid	10	0.016456811	0.02286878	44.02080023
		20	0.016457144	0.022869732	44.01885011
		30	0.016456885	0.022867327	44.02887424
		40	0.016456888	0.022870528	44.01028107
		50	0.016457102	0.022868327	44.02443017

Table 6. GT-VIX-GARCH model performance.

Model	Activation Function	Nodes	MAE	RMSE	MAPE
GT-VIX-GARCH	Relu	10	0.016464618	0.022961953	43.6754142
		20	0.016463169	0.022961503	43.66309023
		30	0.016464239	0.022963376	43.66271481
		40	0.016464096	0.022965466	43.65610286
		50	0.016468159	0.022966994	43.68492065
	Tanh	10	0.016465239	0.022964124	43.67126798
		20	0.016463913	0.022960066	43.67926114
		30	0.016464939	0.022962888	43.67179649
		40	0.01646529	0.022962936	43.68079538
		50	0.016463781	0.022958457	43.68308928
	Elu	10	0.016464796	0.022964955	43.66256389
		20	0.016465883	0.02296502	43.67128545
		30	0.016464635	0.022962084	43.67544449
		40	0.016466452	0.022966505	43.66998111
		50	0.016462585	0.022957731	43.67119374
	Sigmoid	10	0.01646477	0.022963578	43.67003712
		20	0.016461495	0.022957864	43.67131767
		30	0.016464624	0.022961432	43.67831534
		40	0.016464975	0.022965387	43.66149144
		50	0.01647861	0.02302727	43.4274305

Table 7. GT-H model performance.

Model	Activation Function	Nodes	MAE	RMSE	MAPE
GT-H	Relu	10	0.016941584	0.022027929	52.70636488
		20	0.016941599	0.02202816	52.70680583
		30	0.016941163	0.022027678	52.70298093
		40	0.016940914	0.022027074	52.70274993
		50	0.016941279	0.022027853	52.70347102
	Tanh	10	0.016941714	0.022027935	52.70708064
		20	0.016942079	0.022028228	52.70821739
		30	0.016941977	0.022028122	52.70722389
		40	0.016941485	0.022028033	52.70475844
		50	0.016940999	0.022027369	52.70261778
	Elu	10	0.01694181	0.022028066	52.70750404
		20	0.016941503	0.022027574	52.70587155
		30	0.016942019	0.022027821	52.7097488
		40	0.01694203	0.022027968	52.70826475
		50	0.01694185	0.022027961	52.71021157
	Sigmoid	10	0.016941511	0.022027945	52.70543218
		20	0.016941295	0.022027628	52.70568077
		30	0.016941514	0.022028015	52.70581447
		40	0.016942199	0.022028407	52.70889483
		50	0.016941966	0.022027852	52.70746498

Table 8. GT-VIX-H model performance.

Model	Activation Function	Nodes	MAE	RMSE	MAPE
GT-VIX-H	Relu	10	0.016745304	0.022489019	48.02497968
		20	0.01674541	0.02248914	48.0246867
		30	0.016746443	0.022489882	48.02759455
		40	0.016745041	0.022488647	48.02404413
		50	0.016748236	0.022522086	47.82463374
	Tanh	10	0.016745657	0.022489571	48.02533742
		20	0.016745787	0.02248942	48.02683015
		30	0.016745458	0.02248931	48.02374776
		40	0.016744942	0.022489028	48.0238152
		50	0.01674544	0.022489515	48.02256555
	Elu	10	0.01674516	0.02248885	48.02508728
		20	0.016745349	0.022488898	48.02401028
		30	0.016745336	0.022489	48.02517194
		40	0.016745663	0.022488933	48.02556819
		50	0.016745491	0.022489525	48.02760086
	Sigmoid	10	0.016745208	0.022489047	48.02402778
		20	0.016745744	0.022489067	48.02780298
		30	0.0167453	0.022489284	48.0246795
		40	0.016745533	0.022488691	48.02600782
		50	0.016745073	0.022489534	48.02441867

Table 9. GT-VIX-GARCH-H model performance.

Model	Activation Function	Nodes	MAE	RMSE	MAPE
GT-VIX-GARCH-H	Relu	10	0.016099377	0.022304876	41.97126277
		20	0.016101827	0.02228648	42.07658833
		30	0.016095555	0.022284853	42.03342476
		40	0.016109686	0.022277813	42.13001676
		50	0.016097934	0.02228606	42.05669279
	Tanh	10	0.016098348	0.022302633	41.964812188
		20	0.01609808	0.022231812	42.38508374
		30	0.016094577	0.022298402	42.01919185
		40	0.016098404	0.022302775	42.04282541
		50	0.016096921	0.022285756	42.04785915
	Elu	10	0.016108832	0.02236164	41.71558473
		20	0.016100976	0.022290669	42.06674165
		30	0.016098687	0.022285274	42.06192355
		40	0.016101002	0.022291082	42.06727282
		50	0.016105162	0.022295976	42.088699
	Sigmoid	10	0.016099661	0.022292957	42.06001306
		20	0.016099905	0.022281962	42.06856524
		30	0.016100028	0.022278299	42.07671847
		40	0.016105448	0.022285267	42.1902812
		50	0.016096398	0.02229977	42.03760228

Table 10. Input variables of models.

			Input Variables		
Models	LV_t (x_0)	GT (x_1)	VIX (x_2)	GJR-GARCH(1,1) (x_3)	EGARCH(3,3) (x_4)
GT-H model	O	O	X	X	X
GT-VIX-H model	O	O	O	X	X
GT-VIX-GARCH-H model	O	O	O	O	O

Table 9 shows the results of three performance measures obtained by the GT-VIX-GARCH-H model. We can see the improvement in forecasting accuracy in Table 9. The results in Table 9 show that the hybrid models with selected GARCH models based on the HONN model for volatility forecasting of Bitcoin reduce the performance measures (MAE, RMSE, MAPE). That is, in all cases, the measures decrease compared to the measures of the other models. More specifically, compared to the GJR-GARCH(1,1) forecast, MAE is reduced by 11 %, MAPE is reduced by 30 %. Furthermore, we analyze the robustness of our results to determine whether the proposed models are statistically significant. For the analysis, we apply the MCS test [50] to GT-VIX-GARCH-H models. The detailed results of the MCS test, which can be interpreted as a level of confidence for the forecasts, are presented in Table 11. According to the results in Table 11, we can find that the GT-VIX-GARCH-H model with the Relu function and 30 nodes, which has the lowest MAE, is the best model for forecasting of Bitcoin volatility.

Table 11. Model confidence set.

Loss Function Ranking	Model	Activation Function	Nodes	MAE	MCS
1	GT-VIX-GARCH-H	Relu	30	0.016095555	1.000
2	GT-VIX-GARCH-H	Tanh	20	0.01609808	0.991
3	GT-VIX-GARCH-H	Tanh	30	0.016094577	0.991
4	GT-VIX-GARCH-H	Relu	50	0.016097934	0.991
5	GT-VIX-GARCH-H	Tanh	40	0.016098404	0.991

4. Concluding Remarks

We develop the models based on the neural networks for forecasting volatility of Bitcoin price in this paper. Specifically, we propose several hybrid models to improve the forecasting and conduct more than 10,000 experiments to find the optimized model. We investigate as follows. Firstly, we construct the ANN-GARCH models with 1-day lagged volatility, Google Trends, VIX and outputs of GARCH models based on the previous works. Secondly, we propose the new hybrid models which incorporate the outputs of GARCH models as input to HONN model. HONN model, which use the linear combinations of the variables as the input variables, is efficient and performs generally better than the classic ANN mode when the number of good input variables for the ANN model is small. In fact, most of the proposed hybrid models show good performances with no statistical difference, but we focus on finding the best forecasting model for Bitcoin's volatility.

In order to find the best model among the proposed models, we carry out many experiments changing the activation functions and the number of nodes. We also adopt three performance measures to compare the forecasting accuracy of the proposed models. Consequently, the hybrid models based on the HONN model which can capture higher-order correlations in input variables show the improved performance for forecasting of Bitcoin volatility. Compared to the best GARCH model, the best GT-VIX-GARCH-H model improves by 11%, 2.2% and 30% for MAE, RMSE and MAPE, respectively. In addition, compared to the best ANN-GARCH model, the best GT-VIX-GARCH-H model improves by 2.2%, 2.5% and 3.9% for MAE, RMSE and MAPE, respectively. In other words, these results show that the hybrid models based on the HONN model provide more accurate forecasting results and are appropriate for forecasting of volatility in the Bitcoin market.

Author Contributions: G.K. designed the experiments; M.S. collected and analyzed the data; M.S. and G.K. contributed analysis tools; M.S. and G.K. wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Research Foundation of Korea grant funded by the Korea government (No. NRF-2017R1E1A1A03070886).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; ReadLiberty.Org: Chicago, IL, USA, 2008.
2. Urquhart, A. The inefficiency of Bitcoin. *Econ. Lett.* **2016**, *148*, 80–82. [CrossRef]
3. Stavroyiannis, S.; Babalos, V.; Bekiros, S.; Lahmiri, S.; Uddin, G.S. The high frequency multifractal properties of Bitcoin. *Physica A* **2019**, *520*, 62–71. [CrossRef]
4. Gajardo, G.; Kristjanpoller, W.D.; Minutolo, M. Does bitcoin exhibit the same asymmetric multifractal cross-correlations with crude oil, gold and DJIA as the Euro, Great British Pound and Yen. *Chaos Solitons Fractals* **2018**, *109*, 195–205. [CrossRef]
5. Yonghong, J.; He, N.; Weihua, R. Time-varying long-term memory in Bitcoin market. *Financ. Res. Lett.* **2018**, *25*, 280–284.
6. Dyhrberg, A.H. Bitcoin, gold and the dollar–a garch volatility analysis. *Financ. Res. Lett.* **2016**, *16*, 85–92. [CrossRef]
7. Dyhrberg, A.H. Hedging capabilities of bitcoin. is it the virtual gold? *Financ. Res. Lett.* **2016**, *16*, 139–144. [CrossRef]
8. Marie, B.; Kim, O.; Ariane, S. Virtual currency, tangible return: Portfolio diversification with bitcoin. *J. Asset Manag.* **2015**, *16*, 365–373.
9. Hong, K. Bitcoin as an alternative investment vehicle. *Inf. Technol. Manag.* **2017**, *18*, 265–275. [CrossRef]
10. Chuen, D.L.K.; Guo, L.; Wang, Y. Cryptocurrency: A new investment opportunity? *J. Altern. Invest.* **2017**, *20*, 16–40. [CrossRef]
11. Iwamura, M.; Kitamura, Y.; Matsumoto, T.; Saito, K. Can we stabilize the price of a Cryptocurrency?: Understanding the design of Bitcoin and its potential to compete with Central Bank money. *Hitotsub. J. Econ.* **2019**, *60*, 41–60. [CrossRef]

12. Yermack, D. Is Bitcoin a Real Currency? An Economic Appraisal. In *Handbook of Digital Currency*; Academic Press: Cambridge, MA, USA, 2015; pp. 31–44.
13. Baur, D.G.; Hong, K.H.; Lee, A.D. Bitcoin: Medium of exchange or speculative assets? *J. Int. Financ. Mark. Inst. Money* **2018**, *54*, 177–189. [CrossRef]
14. Baur, D.G.; Dimpfl, T. Asymmetric volatility in cryptocurrencies. *Econ. Lett.* **2018**, *173*, 148-151. [CrossRef]
15. Lahmiri, S.; Bekiros, S. Chaos, randomness and multi-fractality in bitcoin market. *Chaos Solitons Fractals* **2018**, *106*, 28–34. [CrossRef]
16. Lahmiri, S.; Bekiros, S.; Salvi, A. Long-range memory, distributional variation and randomness of bitcoin volatility. *Chaos Solitons Fractals* **2018**, *107*, 43–48. [CrossRef]
17. Balcilar, M.; Bouri, E.; Gupta, R.; Roubaud D. Can volume predict Bitcoin returns and volatility? A quantiles-based approach. *Econ. Model* **2017**, *64*, 74–81. [CrossRef]
18. Katsiampa, P. Volatility estimation for bitcoin: A comparison of GARCH models. *Econ. Lett.* **2017**, *158*, 3–6. [CrossRef]
19. Chu, J.; Chan, S.; Nadarajah, S.; Osterrieder, J. GARCH modelling of cryptocurrencies. *J. Risk Financ. Manag* **2017**, *10*, 17. [CrossRef]
20. Conrad, C.; Custovic, A.; Ghysels, E. Long-and Short-Term Cryptocurrency Volatility Components: A GARCH-MIDAS Analysis. *J. Risk Financ. Manag.* **2018**, *11*, 23. [CrossRef]
21. Kristjanpoller, W.; Minutolo, M.C. A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Syst. Appl.* **2018**, *109*, 1–11. [CrossRef]
22. Peng, Y.; Albuquerque, P.H.M.; Sá, J.M.C.; Padula, A.J.A.; Montenegro, M.R. The best of two worlds: Forecasting high frequency volatility for cryptocurrencies and traditional currencies with Support Vector Regression. *Expert Syst. Appl.* **2018**, *97*, 177–192. [CrossRef]
23. Lahmiri, S.; Bekiros, S. Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos Solitons Fractals* **2019**, *118*, 35–40. [CrossRef]
24. Tseng, C.H.; Cheng, S.T.; Wang, Y.H.; Peng, J.T. Artificial neural network model of the hybrid EGARCH volatility of the Taiwan stock index option prices. *Physica A* **2008**, *387*, 3192–3200. [CrossRef]
25. Tseng, C.H.; Cheng, S.T.; Wang, Y.H. New hybrid methodology for stock volatility prediction. *Expert Syst. Appl.* **2009**, *36*, 1833–1839. [CrossRef]
26. Zahedi, J.; Rounaghi, M.M. Application of artificial neural network models and principal component analysis method in predicting stock prices on Tehran Stock Exchange. *Physica A* **2015**, *438*, 178–187. [CrossRef]
27. Kristjanpoller, W.; Minutolo, M.C. Gold price volatility: A forecasting approach using the Artificial neural network-GARCH model. *Expert Syst. Appl.* **2015**, *42*, 7245–7251. [CrossRef]
28. Kristjanpoller, W.; Minutolo, M.C. Forecasting volatility of oil price using an artificial neural network-GARCH model. *Expert Syst. Appl.* **2016**, *65*, 233–241. [CrossRef]
29. Lahmiri, S. Modeling and predicting historical volatility in exchange rate markets. *Physica A* **2017**, *471*, 387–395. [CrossRef]
30. Kristjanpoller, W.; Hernández, E. Volatility of main metals forecasted by a hybrid ANN-GARCH model with regressors. *Expert Syst. Appl.* **2017**, *84*, 290–300. [CrossRef]
31. Hajizadeh, E.; Seifi, A.; Zarandi, M.F.; Turksen, I.B. A hybrid modeling approach for forecasting the volatility of S&P 500 index return. *Expert Syst. Appl.* **2012**, *39*, 431–436.
32. Kristjanpoller, W.; Fadic, A.; Minutolo, M.C. Volatility forecast using hybrid Neural Network models. *Expert Syst. Appl.* **2014**, *41*, 2437–2442. [CrossRef]
33. Lahmiri, S.; Boukadoum, M. An ensemble system based on hybrid EGARCH-ANN with different distributional assumptions to predict S&P500 intraday volatility. *Fluct. Noise Lett.* **2015**, *14*, 1550001.
34. Seo, M.; Lee, S.; Kim, G. Forecasting the Volatility of Stock Market Index Using the Hybrid Models with Google Domestic Trends. *Fluct. Noise Lett.* **2019**, *18*, 1950006. [CrossRef]
35. Engle, R.F. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* **1982**, *50*, 987–1007. [CrossRef]
36. Bollerslev, T. Generalized autoregressive conditional heteroskedasticity. *J. Econom.* **1986**, *31*, 307–327. [CrossRef]
37. Nelson, D.B. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica* **1991**, *59*, 347–370. [CrossRef]

38. Glosten, L.; Jagannathan, R.; Runkle, D. On the relation between the expected value and the volatility nominal excess return on stocks. *J. Financ.* **1993**, *46*, 1779–1801. [CrossRef]
39. Brownlees, C.; Engle, R.; Kelly, B. A practical guide to volatility forecasting through calm and storm. *J. Risk* **2011**, *14*, 3–22. [CrossRef]
40. Giles, L.; Maxwell, T. Learning, invariance and generalization in higher order neural networks. *Appl. Opt.* **1987**, *26*, 4972–4978. [CrossRef]
41. Zhang, M.; Xu, S.; Fulcher, J. Neuron-adaptive higher order neural-network models for automated financial data modelling. *IEEE Trans. Neural Netw.* **2002**, *3*, 188–204. [CrossRef]
42. Xiong, R.; Nichols, E.P.; Shen, Y. Deep learning stock volatility with Google domestic trends. *arXiv* **2015**, arXiv:1512.04916.
43. Hamid, A.; Heiden, M. Forecasting volatility with empirical similarity and Google Trends. *J. Econ. Behav. Organ.* **2015**, *117*, 62–81. [CrossRef]
44. Dimpfl, T.; Jank, S. Can Internet search queries help to predict stock market volatility? *Eur. Financ. Manag.* **2016**, *22*, 171–192. [CrossRef]
45. Kristoufek, L. BitCoin Meets Google Trends and Wikipedia: Quantifying the Relationship between Phenomena of the Internet Era. *Sci. Rep.* **2013**, *3*, 3415. [CrossRef] [PubMed]
46. Kjærland, F.; Khazal, A.; Krogstad, E.A.; Nordstrøm, F.B.; Oust, A. An Analysis of Bitcoin's Price Dynamics. *J. Risk Financ. Manag.* **2018**, *11*, 63. [CrossRef]
47. Aalborg, H.A.; Molnár, P.; Vries, J.E. What can explain the price, volatility and trading volume of Bitcoin? *Financ. Res. Lett.* **2019**, *29*, 255–265. [CrossRef]
48. Botchkarev, A. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv* **2018**, arXiv:1809.03006.
49. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
50. Hansen, P.R.; Lunde, A.; Nason, J.M. The model confidence set. *Econometrica* **2011**, *79*, 453–497. [CrossRef]

applied sciences

MDPI

Article

A Comparison of Time-Series Predictions for Healthcare Emergency Department Indicators and the Impact of COVID-19

Diego Duarte [1,2], Chris Walshaw [2,*] and Nadarajah Ramesh [2]

[1] Transforming Systems, 8th Floor, 6 Mitre Passage, London SE10 0ER, UK; Diego.Duarte@vitalhub.com
[2] School of Computing & Mathematical Sciences, University of Greenwich, Park Row, Greenwich, London SE10 9LS, UK; n.i.ramesh@gre.ac.uk
* Correspondence: c.walshaw@gre.ac.uk; Tel.: +44-20-8331-8142

Featured Application: This application is being developed as part of a suite of tools used by the National Health Service (NHS) to analyse and predict pressure in resource management indicators (see transformingsystems.com for more details).

Abstract: Across the world, healthcare systems are under stress and this has been hugely exacerbated by the COVID pandemic. Key Performance Indicators (KPIs), usually in the form of time-series data, are used to help manage that stress. Making reliable predictions of these indicators, particularly for emergency departments (ED), can facilitate acute unit planning, enhance quality of care and optimise resources. This motivates models that can forecast relevant KPIs and this paper addresses that need by comparing the Autoregressive Integrated Moving Average (ARIMA) method, a purely statistical model, to Prophet, a decomposable forecasting model based on trend, seasonality and holidays variables, and to the General Regression Neural Network (GRNN), a machine learning model. The dataset analysed is formed of four hourly valued indicators from a UK hospital: Patients in Department; Number of Attendances; Unallocated Patients with a DTA (Decision to Admit); Medically Fit for Discharge. Typically, the data exhibit regular patterns and seasonal trends and can be impacted by external factors such as the weather or major incidents. The COVID pandemic is an extreme instance of the latter and the behaviour of sample data changed dramatically. The capacity to quickly adapt to these changes is crucial and is a factor that shows better results for GRNN in both accuracy and reliability.

Keywords: healthcare; COVID; time-series predictions; machine learning; ARIMA; Prophet; GRNN

Citation: Duarte, D.; Walshaw, C.; Ramesh, N. A Comparison of Time-Series Predictions for Healthcare Emergency Department Indicators and the Impact of COVID-19. *Appl. Sci.* **2021**, *11*, 3561. https://doi.org/10.3390/app11083561

Academic Editors: Anton Civit and Grzegorz Dudek

Received: 5 November 2020
Accepted: 25 March 2021
Published: 15 April 2021

1. Introduction

Across the world, healthcare systems are under stress and this has been hugely exacerbated by the COVID pandemic. Key Performance Indicators (KPIs), usually in the form of time-series data, are used to help manage that stress. Making reliable predictions of these indicators, particularly for emergency departments (ED), can help to identify pressure points in advance and also allows for scenario planning, for example, to optimise staff shifts and planning escalation actions.

According to Medway Foundation Trust (MFT), where the results of this study are applied, it is important to accurately forecast after exceptional events in their data, such as the pandemic, because forecasts are of increased importance at these critical moments. When the uncertainty level is greater, correct predictions may benefit their decision making more than usual (although models must also perform well in non-exceptional circumstances).

When analysing the healthcare systems, great significance has been placed on predicting patient arrivals in acute units, and in particular emergency department (ED) attendances and throughput. Typically, patients arrive at irregular intervals, often beyond the control of the hospital, and arrivals show strong seasonal and stochastic fluctuations driven by factors such as weather, disease outbreaks, day of the week and socio-demographic effects [1]. Accordingly, researchers use a variety of methods to predict ED visits over various periodic intervals, e.g., [2,3].

Furthermore, predicting KPIs in EDs can depend on several factors, e.g., [4,5]. Typically, EDs deal with a variety of life-threatening emergencies, arising from causes such as car accidents, disease, pollution, work-place hazards, aging, changing population, pandemics and many other sources. However, multivariate models usually consider only a small number of dependent/independent variables out of numerous possibilities, and in ED that can result in dangerous omissions. For example, variables that are not captured (especially those with high correlations, multi-variable correlation lags and high levels of noise) may drive multi-variate predictions of captured variables in the wrong direction [6].

For predictions of this type, a common approach is to auto-correlate the data. Since all the numerous variables have been an influence on the past values of the time-series under investigation, prediction models can use these data as a baseline to calculate future time-series values. Apart from this baseline, other variables, both dependent and independent, and represented by other indicators, may be included in the model by considering the covariance between them.

Accurate and reliable forecasts of various types of indicator can help to efficiently allocate key healthcare resources, including staff, equipment and vehicles, when and where they are most needed [2], and avoid allocating them at non-critical periods. This justifies considerable efforts to make reliable and accurate predictions of ED data to help hospital managers in making decisions about how to meet the expected healthcare demand effectively and in a timely fashion [1].

As a linear model, ARIMA can typically capture linear patterns in a time series efficiently, and so many studies adopt it either to evaluate relationships between variables, or use ARIMA forecasts as a benchmark against which to test the effectiveness of other models, e.g., [7].

This has been applied to ED data previously and as examples, Sun et al. [3], Afilal et al. [8], and Milner [9] have developed ARIMA models for forecasting hospital ED attendances and have verified the fitness of ARIMA, in that context, as a readily available tool for predicting ED workload. In particular, Milner [9] used ARIMA to forecast yearly patient attendances at EDs in the Trent region of the UK and found that the forecast for attendances was just 3 per cent away from the actual figure. Meanwhile, Sun et al. [3] were able to improve on the ARIMA results for daily patient attendances in Singapore General Hospital by incorporating other variables such as public holidays, periodicities and air quality.

Another option is to use proprietary models. One such, Prophet, is a forecasting model developed at Facebook Research, an R&D branch of the company. Taylor & Letham [10] observed that ARIMA forecasts can be prone to large trend errors, particularly when a change in trend occurs close to the cut-off period, and often fail to capture any seasonality. In experiments with Prophet, Taylor and Letham [10] predicted the daily number of Facebook events in a 30-day horizon with 25% more accuracy than ARIMA. The accurate prediction of these daily Facebook events, which are captured by an indicator with similar characteristics as some of the healthcare indicators, particularly in terms of in trend, seasonality and holidays, and which are at the heart of the Prophet model decomposition, motivated their use in this study.

Finally, a number of Machine Learning models are used to predict data, specifically for time-series processes, and in particular Neural Network variants have presented encouraging results for classification problems, e.g., [11,12]. In this study, after fitting models of LSTM (Long Short-Term Memory), RNN (Recurrent Neural Network) and RBN (Radial Basis Network), we present the results for the model found to best fit the data sample in our simulations, that is RBN, and in particular GRNN (Generalised Regression Neural Networks).

The rest of the paper is organised as follows: Sections 2–4 give more details of ARIMA, Prophet and GRNN, respectively. Section 5 describes the data and Section 6 gives details of how COVID has impacted on it. The main parts of the paper are Section 7 which describes the experimentation, and Section 8 that analyses those results. Finally, Section 9 presents conclusions and suggestions for further work.

2. Autoregressive Integrated Moving Average (ARIMA)

The Autoregressive Integrated Moving Average (ARIMA) model integrates Auto Regressive and Moving Average calculations. A basic requirement for predicting time-series with ARIMA is that the time-series should be stationary or, at the very least, trend-stationary [3,7]. A stationary series is one that has no trend, and where variations around the mean have a constant amplitude, e.g., [1,13,14]. Although ARIMA expects a stationary stochastic process as input, very few datasets are natively in such format, thus the use of differencing to "stationarise" is in the model identification stage [1,7] (Figure 1).

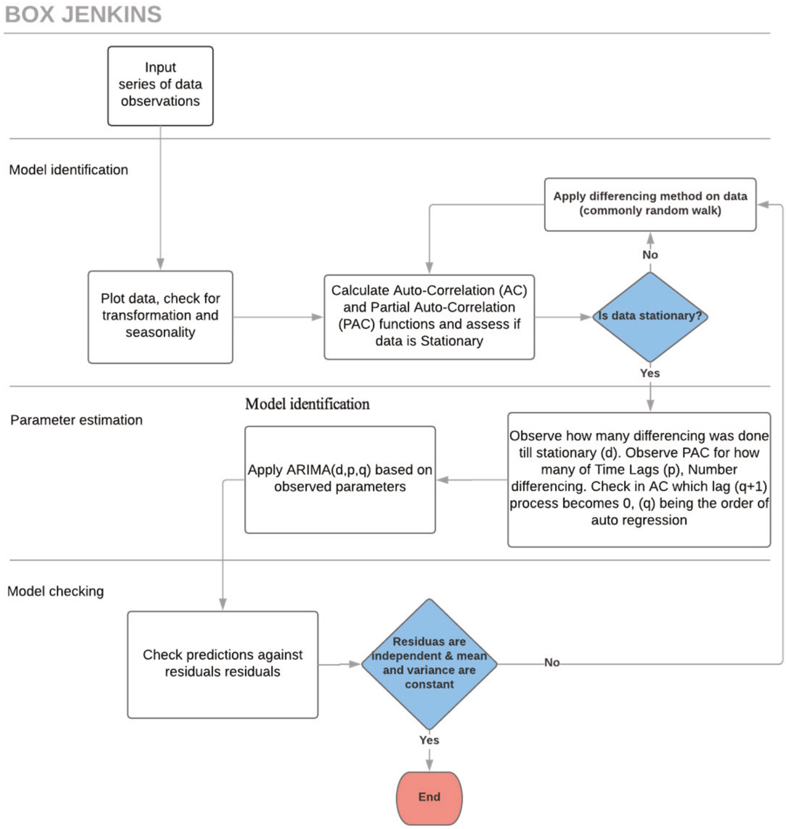

BOX JENKINS
Input series of data observations
Model identification
Apply differencing method on data (commonly random walk)
No
Plot data, check for transformation and seasonality
Calculate Auto-Correlation (AC) and Partial Auto-Correlation (PAC) functions and assess if data is Stationary
Is data stationary?
Yes
Parameter estimation
Model identification
Apply ARIMA(d,p,q) based on observed parameters
Observe how many differencing was done till stationary (d). Observe PAC for how many of Time Lags (p), Number differencing. Check in AC which lag (q+1) process becomes 0, (q) being the order of auto regression
Model checking
Check predictions against residuals residuals
Residuas are independent & mean and variance are constant
No
Yes
End

Figure 1. The Box-Jenkins/ARIMA iterative fitting process.

The ARIMA model has 3 key parameters (p, d, q), all non-negative integers: p is the order (the number of time lags) of the autoregressive model, d is the degree of differencing (the number of times the data have had past values subtracted if random walk is chosen), and q is the order of the moving-average model. The values for p, d, and q are defined by calculations and subsequent analysis and the process of fitting an ARIMA model, by calculating these parameters, is commonly known as the Box-Jenkins method [15].

The integration of the p autoregressive and q moving average terms, gives rise to the formula [15]:

$$\hat{y} = c + \epsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-1}$$

where c represents a constant variable, error terms ϵ_t are generally assumed to be independent, uniformly distributed random variables, y_t are previous observed values of the time-series and ϕ_i and θ_i are coefficients determined as part of the process.

Each indicator under analysis here has been through the three-stage ARIMA/Box-Jenkins iterative modelling approach (Figure 1):

1. Model identification, to ensure that variables are stationary: if not, differencing methods such as random walk are applied, any seasonality is identified, and then AC (Auto Correlation) and PAC (Partial Auto Correlation) functions are plotted [16].
2. Parameter estimation: using methods such as Non-linear Least Squares Estimation or Maximum Likelihood Estimation together with the outcome of the AC and PAC functions [16].
3. Model checking, to test whether predictions now adhere to the requirements of a stationary univariate process: in particular, residuals must be independent of each other and within an acceptable range. Additionally, the mean and variance should be constant over the time. If the model is still inadequate, the iterative process returns to stage 1 and an alternative method for differencing is applied.

3. Prophet

Prophet is a forecasting model from Facebook Research. As discussed by Taylor & Letham, forecasting is a data science task, central to many activities within a large organization such as Facebook, and crucial for capacity planning and the efficient allocation of resources [10].

Prophet is motivated by the idea that not every prediction problem can be tackled using the same solution, and has been created with the aim of optimising business forecasting tasks encountered at Facebook that typically have some or all of the following characteristics [10]:

- Regular (hourly, daily, or weekly observations with at least a few months and preferably a year or more of history;
- Strong repeated "human-scale" seasonality (e.g., day of week or time of year);
- Significant holidays, often that occur at irregular intervals, but known in advance (e.g., Thanksgiving, the Super Bowl);
- Not too many missing observations or significant outliers;
- Historical trend changes (e.g., as a result of product launches or changes in logging procedures);
- Trends that include non-linear growth curves, e.g., natural limits or saturation levels.
- Unlike ARIMA models, the measurements do not need to be regularly spaced (stochastic process), thus Prophet also does not require time-series to be stationary.

Prophet uses a composite time-series model to predict $y(t)$, with three main component, trend, seasonality, and holidays, that are combined as:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

Here, $g(t)$ is the trend function, $s(t)$ models seasonal changes, and $h(t)$ represents holidays, whilst ε_t represents an error term which is expected to be normally distributed [10].

3.1. Trend Model

Prophet's main forecast component is the trend term, $g(t)$, which defines how the time-series has developed previously and how it is expected to continue. Two choices are available for this trend function depending on the data characteristics: a Nonlinear

Saturating Growth model, where the growth is non-linear and expected to saturate at a carrying capacity, and a Linear Model, where the rate of growth is stable [17].

In the case of the experiments in Section 7, the choice of trend function was determined automatically by Prophet.

3.2. Seasonality

Prophet also considers seasonal data patterns arising from similar behaviour repeated over several data intervals. To address this component of the model Prophet employs Fourier series' to capture and model periodic effects [17].

This is very appropriate for the data under investigation which may exhibit multiple seasonal patterns. For example, hospital ED departments are typically busier every day between 5 p.m. and 8 p.m. from a variety of causes such as people suffering injuries when commuting, taking relatives to hospital after working hours, or injuries when exercising, etc. Meanwhile longer nested trends can arise as hospital EDs are usually busier in winter as compared with summer due to respiratory illness, slippery conditions, etc.

3.3. Holidays and Events

Holidays and special events provide a relatively significant, and normally predictable changes in time-series. Normally these do not have a regular periodic pattern (unlike, say, weekends) and thus the effects are not well modelled by a smooth cycle.

To address this component Prophet offers the functionality to include a custom list of holidays and events, both past and future [10]. However, this feature has not been tested in the investigations in Section 7.

4. General Regression Neural Network (GRNN)

The General Regression Neural Network (GRNN) proposed by Specht [18] is a feed-forward neural network, responding to an input pattern by processing the input data from one layer to the next with no feedback paths. GRNN is a one-pass learning algorithm with a highly parallel structure. Even in the case a time-series has sparse observations with a non-stochastic process, the model outputs smooth transitions between resulting observations. The algorithmic form may be used for any regression problem provided that linearity is not justified.

The GRNN is considered a type of RBF (Radial Basis Function) neural network, that employs a fast, single pass learning. It consists of a hidden layer with RBF neurons. Typically, the hidden (Pattern) layer has the number of neurons similar to training examples. The center of a neuron is the linked training example, thus the output provides a measure of the closeness of the input vector to the training example. A subsequent summation layer is added to compute the results (Figure 2).

Normally, a neuron uses the multivariate Gaussian function [19].

$$\mathrm{G}(x, \mathrm{x_i}) = \exp\left(-\frac{||x - \mathrm{x_i}||^2}{2\sigma^2}\right)$$

where $\mathrm{x_i}$ is the centre, σ the smoothing parameter and x the input vector.

Considering a training set of size n, patterns $\{\mathrm{x_1} \ldots \mathrm{x_n}\}$, and their associated targets, $\{\mathrm{y_1} \ldots \mathrm{y_n}\}$, the output is calculated in 2 stages, first the hidden layer produces weights w_i representing levels of similarity of $\mathrm{x_i}$ to training patterns:

$$w_i = \frac{\exp\left(-\frac{||x-\mathrm{x_i}||^2}{2\sigma^2}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{||x-\mathrm{x_j}||^2}{2\sigma^2}\right)}$$

The further away the training pattern is, the smaller the effect in the weight. The total sum of weights corresponds to 1, representing the proportional strength in the result.

The smoothing parameter σ, provides control on how many parameters are to be considered in the calculation and is an important part of model fitting, as it depends on the level of correlation and lags.

The second stage is the resulting calculation of future values $y(t)$:

$$y(t) = \sum_{i=1}^{n} w_i y_i$$

The resulting calculation is nothing more than a weighted average of training targets.

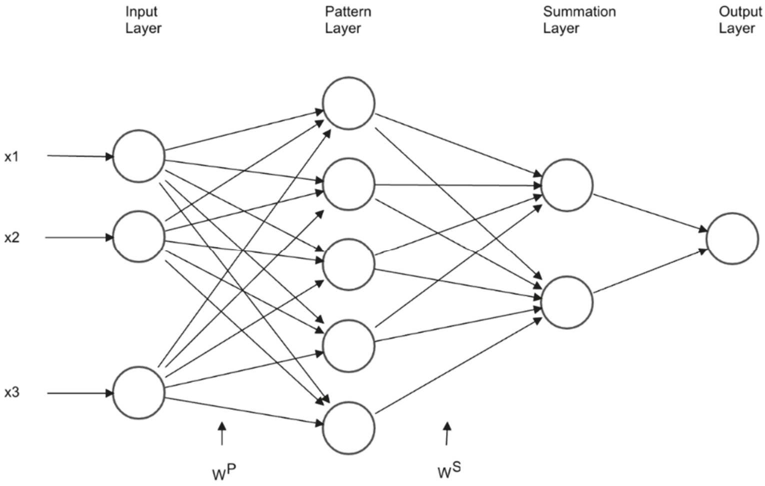


Figure 2. Schematic diagram of generalized regression neural networks (GRNN).

4.1. Multi-Step Ahead Strategy

When predicting an hourly time-series of ED data, such as the Number of Patient Attendances, it is fair to expect to forecast a number of hours ahead, ideally every hour of the next seven days. For this a multi-step ahead strategy must be employed. There are two options, the MIMO (Multiple Input Multiple Output) strategy, and a Recursive strategy.

The MIMO strategy for predicting a series of future observations consists in employing training targets vectors with consecutive observations of the time-series, with the size of those vectors corresponding to the number of predicted observations desired.

The Recursive strategy is based in the one step ahead model as provided by the above equations. The model is applied recursively, matching the desired number of steps ahead, feeding predicted values as input variables. The recursive strategy was the option utilised in the simulations of GRNN in this study: the motivation was not to depend on training target vectors, as this can potentially demand frequent re-training.

4.2. Controlling Seasonality by Autoregressive Lags

The GRNN model does not contain an embedded seasonal parameter, but this can be addressed by the use of autoregressive lags, grouping correlated data into the training vector. The lag definition and the smoothing parameters can be indicated by the AC (Auto Correlation) and PAC (Partial Auto Correlation) functions as discussed in Section 2.

5. Data Description

The choice of the most valuable Emergency Department Key Performance Indicators (ED KPIs), with the intention of capturing pressure in an acute unit, is an inter-disciplinary issue. Input from healthcare professionals, particularly hospital managers, and data scientists, was sought for this study. However, the KPIs must be chosen from data that are already being captured and not every theoretically useful indicator may be readily available.

The four KPIs under analysis, chosen from those available, are shown in Table 1. These data have been provided by Medway Foundation Trust (MFT), located in Kent, South East England and, in particular, focuses on the MFT Hospital ED Acute Unit. The time-series frequency of all 4 the KPIs is hourly.

Table 1. Key Performance Indicator Descriptions.

Key Performance Indicator	Abbreviation	Description
Patients in Department	PD	Hourly average number of patients waiting to be seen in the ED (hourly)
Patient Attendances	PA	Hourly total number of patients with major injuries attending the ED
Unallocated Patients with DTA	UP	Number of patients that have been attended, not discharged, but waiting to be admitted to the hospital, thus occupying resources
Medically Fit for Discharge	MD	Total number of patients who have attended, and do not need further treatment

KPIs of this nature are key metrics in evaluating NHS performance and related KPIs used as targets. For example, the NHS Constitution sets out that a minimum of 95% of patients attending an A&E department should be admitted, transferred or discharged within 4 h of their arrival [20].

6. COVID Impact

The COVID pandemic has impacted UK society, and hence the KPI data, dramatically. The first government advice on social distancing was published on 12 March 2020, before a formal lockdown was announced on 23 March 2020, led to a huge reduction in consumer demand in certain sectors, closure of business and factories and disrupted supply chains [21].

In preparation for the pandemic, hospitals invested in expanding resources that were necessary to treat a high volume of patients with the infection, including respirators, PPE (Personal Protective Equipment), staff, protocols and treatments [22].

During March, NHS trusts rapidly re-designed their services on a large scale to release capacity for treating patients with COVID-19. This included discharging thousands to free up beds, postponing planned treatment, shifting appointments online where possible and redeploying staff, a process covered widely in the media. NHS England alone published more than 50 sets of guidance to hospital specialists for the treatment of non-COVID-19 patients during the pandemic [22].

The impact of such measures was easily observed in the data, simulations and results of this study. Due to the fact of the lockdown added to the fear of contracting COVID, the chosen KPIs had a sudden and dramatic reductions in absolute numbers, as fewer were going to the ED.

Not all prediction models reviewed in this paper were able to immediately account for the sudden changes of this nature and hence their accuracy and reliability dropped drastically. In particular immediately obvious inaccuracies in the models are due to the fact they mostly rely on auto-regressive and seasonal parameters of 6 to 8 weeks prior to the current time. Thus, it usually took 3 to 6 weeks for the models to learn and adapt to the change depending on the indicator. However, well-trained machine learning models present encouraging results, as can be seen below.

7. Simulations and Results

Simulations with the three presented forecasting models have been performed to compare the relative accuracy and reliability when applied to the four chosen indicators. ARIMA, traditionally the standard model for healthcare time-series predictions, is used as a benchmark.

Observing the ACF (Auto Correlation Function) and initial model training, a strong correlation was found by hour and weekday, so data have been classified by these parameters when applying the models. Further to the classification, it was observed that the autocorrelation of the data was stronger in the 8 weeks prior to the forecasting, as the residuals when utilizing this interval were the smallest in simulations. Summarizing, each prediction $\hat{y}$ was calculated using a rolling actuals timeseries subset of the 8 previous observations, classified with a lag = 168.

The subset of 8 observations means that when applying GRNN, the size of the input layer was equal to 8, although the number is small, it presented smaller residuals than trials with sizes of input layer ranging from 12, 20 and 30, this is believed to be due to the higher correlation of the data observed in the ACF. The sigma parameter for the GRNN was chosen to be 0.3, which was found to be the best fit in preliminary experimentation.

Each model has been trained until its residuals were minimal in simulations, then parameters fixed and subsequently applied to the same rolling actuals time-series subset, of the same data source, with equal periods and frequency. Predictions were then compared with actual values and analysed for accuracy and reliability over the period from 1 January 2020 to the end of October 2020. Each prediction made is hourly and has an event horizon of h = 168 h, corresponding to predicting every hour for 7 days in advance.

The results are presented in a number of figures showing actual observed values, predictions and residuals and then analysed in Section 8.

7.1. Patients in Department

Figure 3 shows a comparison of actual observed data (in black) for the Patients in Department indicator compared with the ARIMA (green), Prophet (yellow) and GRNN (purple) predictions. Here, the red coloured square indicates the start of lockdown due to the COVID-19 pandemic. Three regions of the chart can be clearly distinguished (to the left, inside and to the right of the square) and indicate the normal number of Patients per hour in the Medway Foundation Trust ED before, during and after the lockdown, the sudden change, and the way data are slowly growing back in an apparent linear trend, from May to June. It is particularly interesting to see that ARIMA and Prophet overestimate their predictions for a five-week period, whilst GRNN is quick to adjust to the data ingress change. Another interesting observation is that the variance of the data is reduced after the lockdown, making the data easier to predict for all models. This is confirmed in the following figures which show predictions and actuals (top) and residuals (bottom) in more detail for each model.

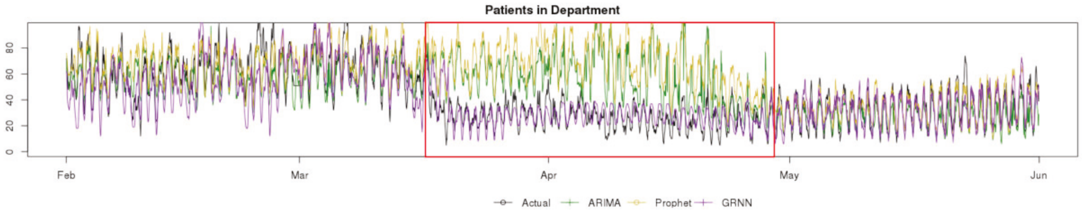


Figure 3. Comparison of Actuals, ARIMA, Prophet and GRNN, February 2020–June 2020.

In Figure 4 it can be observed that the ARIMA model predicts the Patients in Department time-series with a fair accuracy until the lockdown period, where the residuals negatively increase. Then, after five weeks the model learns the new levels and the residuals are again back to acceptable levels.

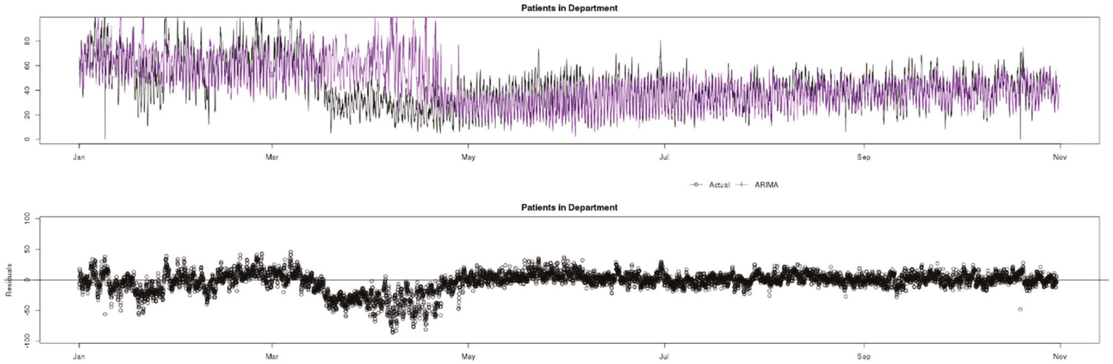


Figure 4. Comparison of Actual and ARIMA with residual distribution chart, January 2020–November 2020.

In Figure 5 it can be observed that Prophet, similar to ARIMA, predicts the Patients in Department indicator with fair accuracy until lockdown, during which the predictions are overestimated and the residuals increase by an even greater margin than ARIMA. However, after five weeks the model learns the new levels and the residuals are again back to acceptable levels.

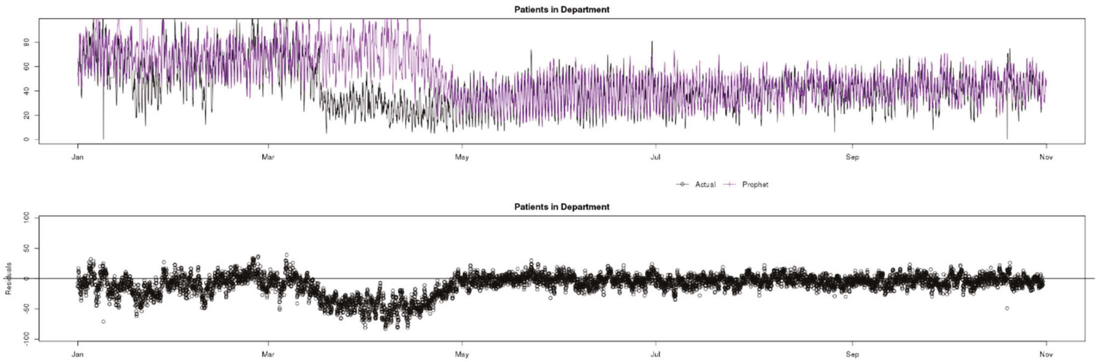


Figure 5. Comparison of Actual and Prophet with residual distribution chart, January 2020–November 2020.

In Figure 6 it can be observed that GRNN, in contrast to the ARIMA and Prophet models, is able to quickly adapt to the change in observed values of Patients in Department due to COVID-19, keeping a good level of accuracy throughout the whole data sample.

7.2. Patient Attendances

Figure 7 shows a comparison of actual (black) data compared to ARIMA (green), Prophet (yellow) and GRNN (purple) predictions for the Patient Attendances indicator. The observed data contain a greater variance than Patients in Department, and this can alter the performance of prediction models. The COVID-19 pandemic caused a similar change in data behaviour to Patient Attendances, and GRNN was also the only model that quickly adapted to the change and accurately predicted the timeseries variables during this period.

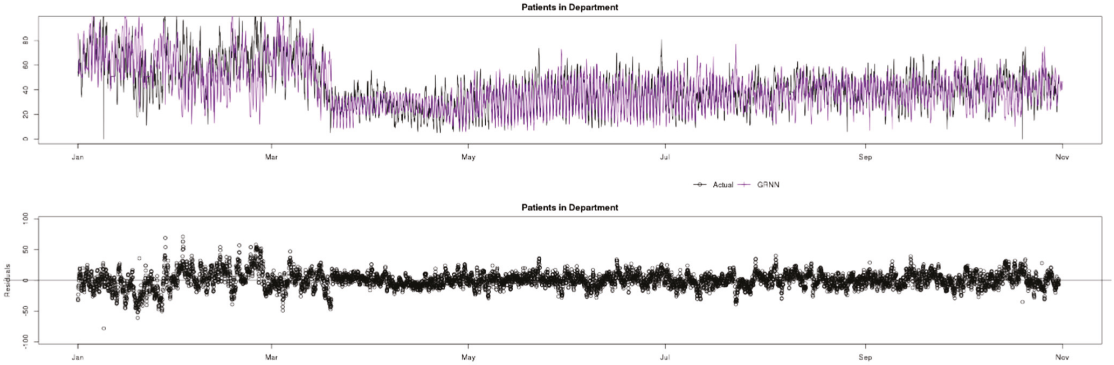


Figure 6. Comparison of Actual and GRNN with residual distribution chart, January 2020–November 2020.

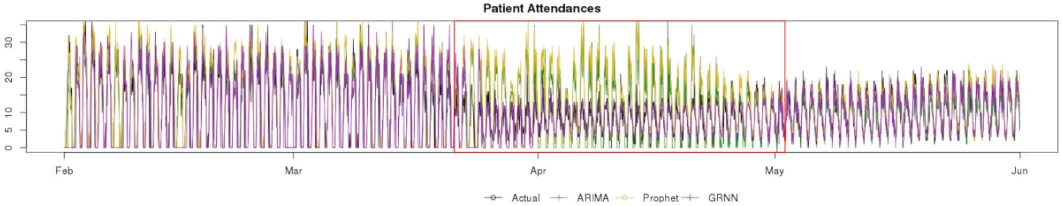


Figure 7. Comparison of Actuals, ARIMA, Prophet and GRNN, February 2020–June 2020.

The following figures breakdown this information and show the residual distributions.

Observing the residuals charts at the bottom of Figures 8–10, in the pre-lockdown phase ARIMA appears to be the best fit model as the residuals are concentrated closer to the x axis, whilst GRNN appears to have greater residuals and less accuracy in this phase. However, when the pandemic changed the data, GRNN was the only one of the three models to quickly adapt and predict with accuracy almost instantaneously, whilst both ARIMA and Prophet overestimated for approximately a 5 weeks period. In the post-lockdown phase, after all three models learned the new data behaviour, the performance is similar.

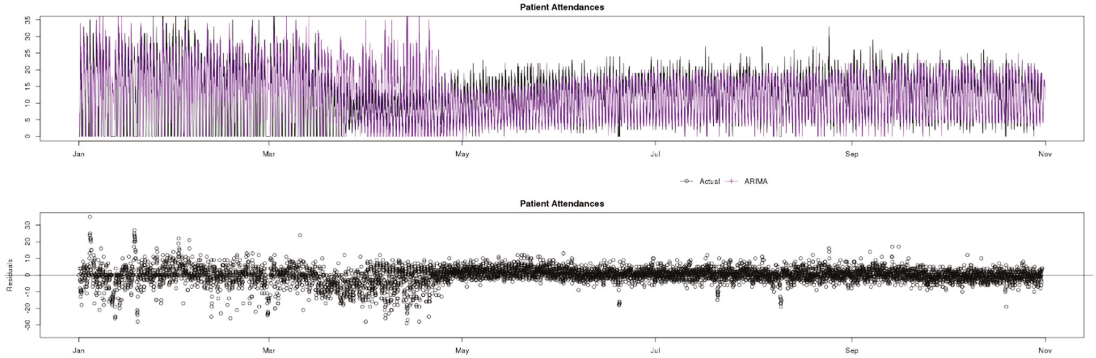


Figure 8. Comparison of Actual and ARIMA with residual distribution chart, January 2020–November 2020.

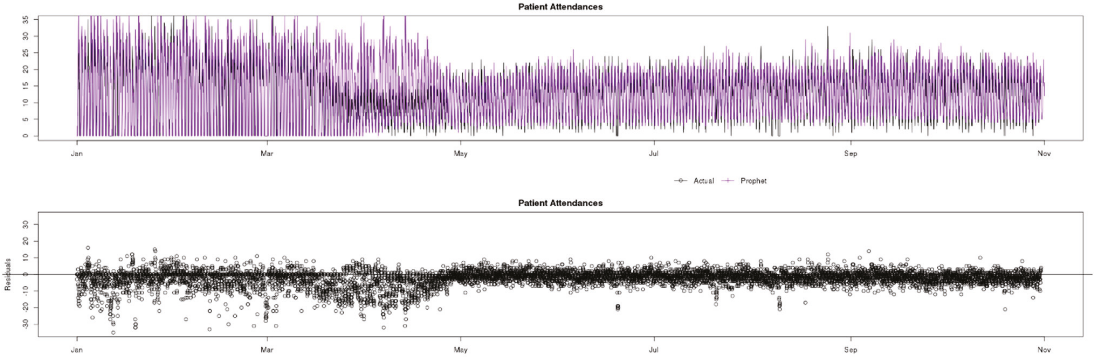


Figure 9. Comparison of Actual and Prophet with residual distribution chart, January 2020–November 2020.

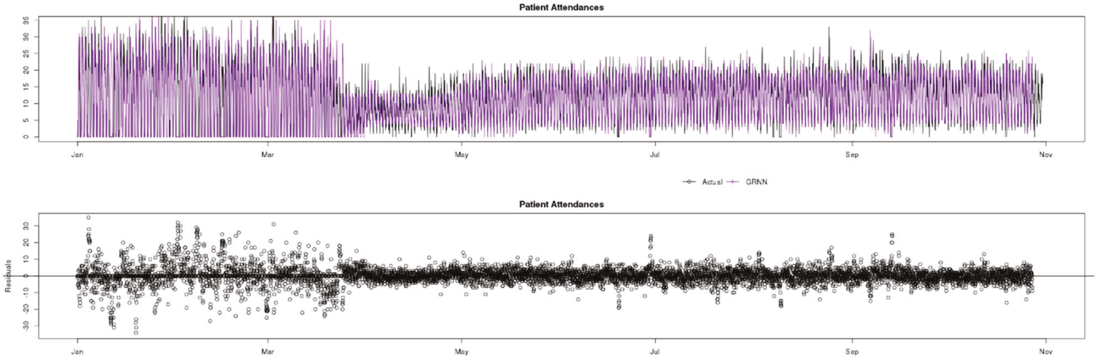


Figure 10. Comparison of Actual and GRNN with residual distribution chart, January 2020–November 2020.

7.3. Unallocated Patients with DTA

Figure 11 shows a comparison of actual (black) data compared to ARIMA (green), Prophet (yellow) and GRNN (purple) predictions for the Unallocated Patients with DTA indicator. This indicator is perhaps the most challenging to predict in this study, as it includes a native and "variable" seasonal factor in that typically the decisions to admit patients are made in specific periods of the day, i.e., at the beginning of the doctors' shifts, after their first meetings with patients. As doctors' shifts commence at different times (unknown by the models), the lag factor of the seasonality is not constant, making the indicator somewhat unpredictable. This is the main reason residuals are typically greater than for other indicators.

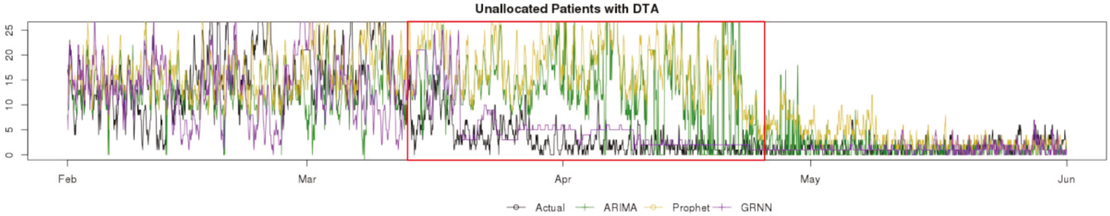


Figure 11. Comparison of Actuals, ARIMA, Prophet and GRNN, February 2020–June 2020.

Looking at the residuals charts at the bottom of Figures 12–14, the unpredictability mentioned can be clearly be observed by the high residuals in the pre-COVID phase for all three analysed models (ARIMA, Prophet and GRNN). During the lockdown affected period, as expected only GRNN predicts values reliably. In the post-lockdown learning period, all three models are able to reliably predict Unallocated Patients with DTA similarly, until October, when data change behaviour again, and ARIMA seems to explain the variables better.

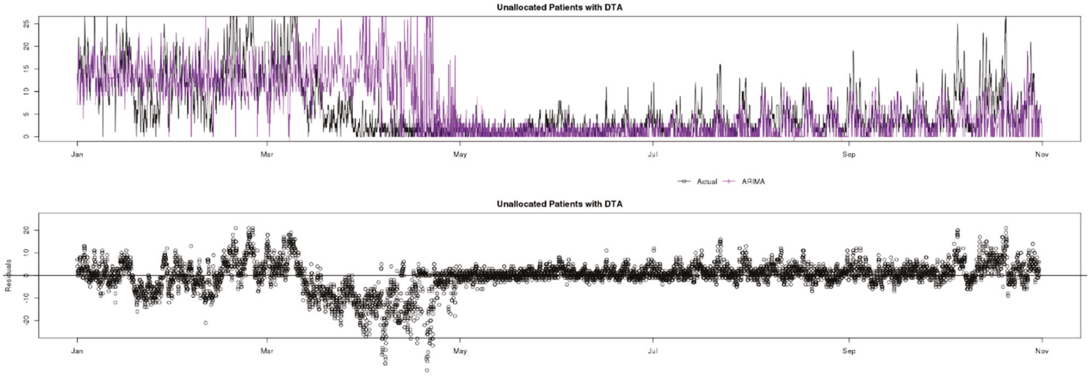


Figure 12. Comparison of Actual and ARIMA with residual distribution chart, January 2020–November 2020.

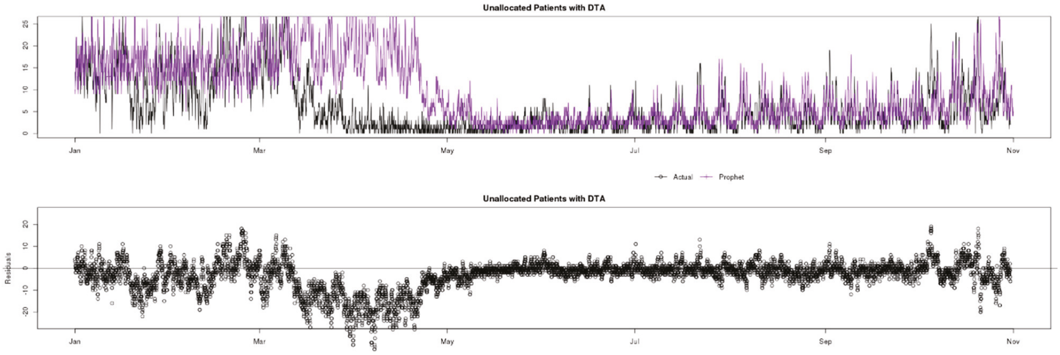


Figure 13. Comparison of Actual and Prophet with residual distribution chart, January 2020–November 2020.

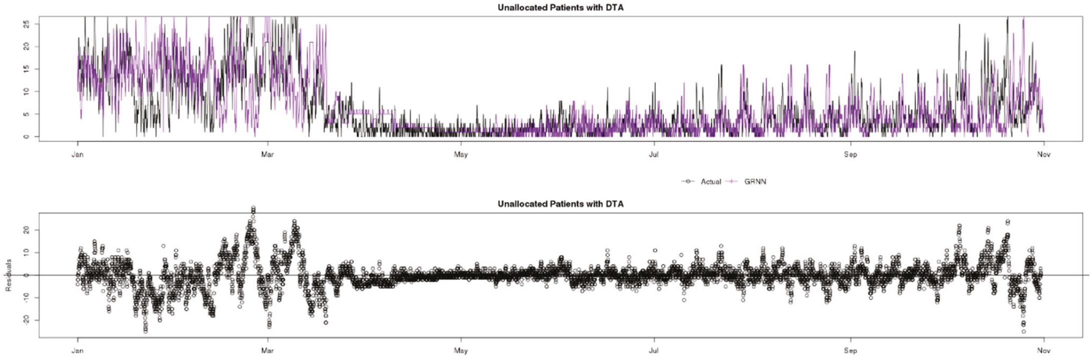


Figure 14. Comparison of Actual and GRNN with residual distribution chart, January 2020–November 2020.

7.4. Medically Fit for Discharge

Figure 15 shows a comparison of actual (black) data compared to ARIMA (green), Prophet (yellow) and GRNN (purple) predictions for the Medically Fit for Discharge indicator. This indicator, unlike Unallocated Patients with DTA presents a very uniform seasonality and is thus a candidate for higher accuracy and reliability predictions. The reason is that doctors typically discharge patients at the same fixed times every day and this creates a very uniform lag factor, that is important to all auto-regressive processes. However, once again in the red square, it can be observed that GRNN is the only prediction model that can quickly adapt and provide reliable predictions during the lockdown period. Finally, there are some obvious outliers in the data, especially in February, but investigation has shown that these are due to data quality issues rather than unusual occurrences.

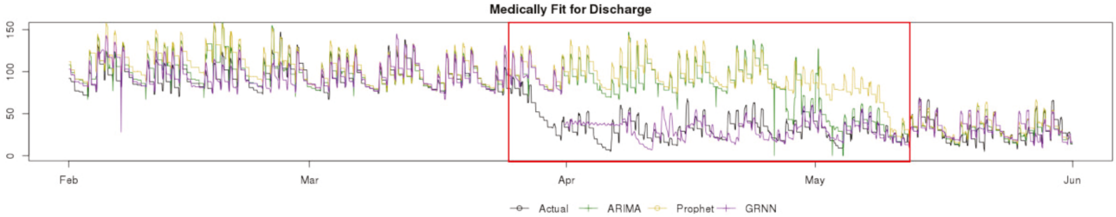


Figure 15. Comparison of Actuals, ARIMA, Prophet and GRNN, February 2020–June 2020.

Once again, the residual charts in Figures 16–18 show very small residuals for all three models, apart from the lockdown period, where GRNN is the only model that can adapt rapidly. This clear accuracy advantage is also highlighted by the red squares in Figures 3, 7, 11 and 15. This may be dependent on the training, especially the size of the input layer, which, although the same for all models (8), by being relatively small may benefit GRNN, when the data change behaviour, and allow it a more agile response.

Finally, in the prediction chart of Figure 18, a few GRNN predictions are missing, and this is due to the model being sensitive to missing input of actual observations in this indicator time-series.

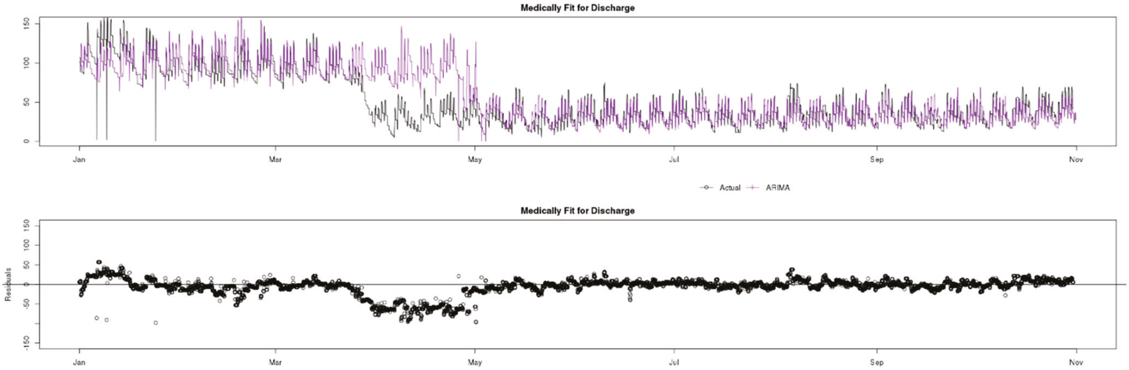


Figure 16. Comparison of Actual and ARIMA with residual distribution chart, January 2020–November 2020.

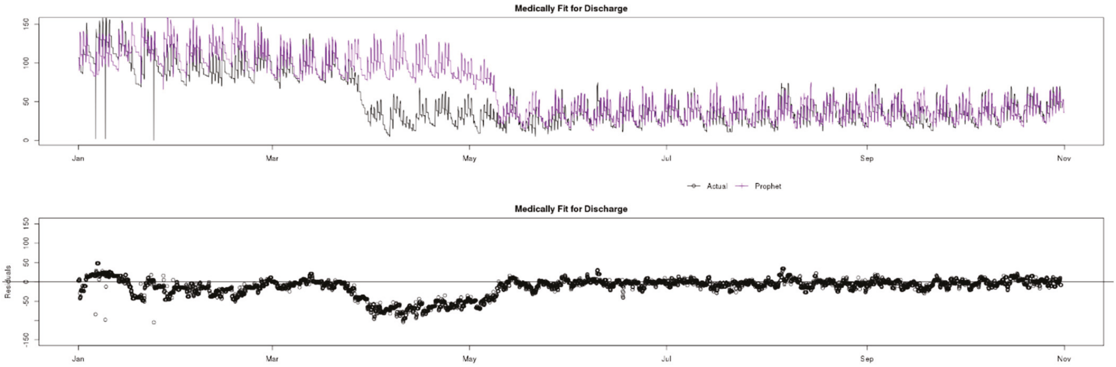


Figure 17. Comparison of Actual and Prophet with residual distribution chart, January 2020–November 2020.

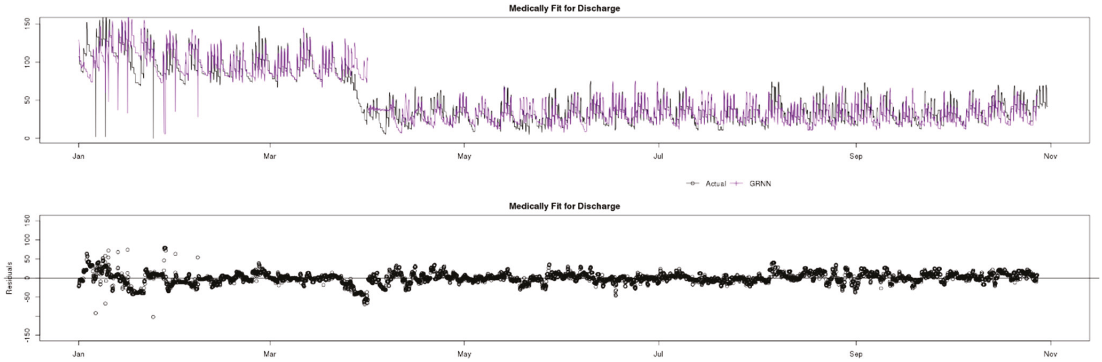


Figure 18. Comparison of Actual and GRNN with residual distribution chart, January 2020–November 2020.

8. Reliability and Accuracy Analysis

Time-series predictions are commonly evaluated by looking at the "residuals", as shown above, which are typically measured as the difference between actual values and predictions, often using the root mean squared error (RMSE).

8.1. Analysis

When emphasizing the importance of relevant measurements in evaluation of forecasting models, Krollner et al., 2010 [23] stated that over 80% of the papers reported that their model outperformed the benchmark model. However, most analysed studies do not consider real world constraints like trading costs and slippage. In this study, for the data presented, the residual analysis has been under discussion, as the usual measurements do not necessarily capture the quality of the time series forecasting in an Emergency Department if considering the applied science aspect, with real-world constraints. In particular, when analysing the environment of an ED, it is often more important to have approximated values constantly close, or within a certain threshold of the actuals, rather than keeping a low perceptual average RMSE.

To understand this, consider a situation where the average RMSE is low but exhibits occasional spikes. In an ED situation a spike corresponds to high unpredicted demand, such as a sudden large influx of patients, and may have dangerous clinical outcomes for patients if resource capacity is significantly exceeded on those occasions. A prediction which is less accurate on average, but which exhibits fewer spikes in RMSE is much preferred as it predicts changing demand more reliably, even if it is not always exact.

For this reason, in the analysis below (Table 2) every simulation contains confidence bands and accuracy metrics which provide a more readable measure for healthcare professionals (following discussions) to compare predictions. Thus "Very good" predictions have residuals that are no further than 15% of the actual mean away from the actual counterpart, "Good" are within 15–25%, "Regular" are within 25–40% of the mean, and any prediction with residual above 40% of the actual mean is considered "Unreliable". Accordingly, all predictions that turned out to be within 40% of the actual values are classed as "Reliable" and overall "Reliability" (i.e., the total percentage of "Very good", "Good" and "Regular" predictions) is used as a summary metric.

Table 2. Reliability comparison of ARIMA, Prophet and GRNN.

Confidence Bands Data Distribution				
	Residual Quality	***ARIMA***	***Prophet***	***GRNN***
Patients in Department	Very good	48.8%	43.5%	48.9%
	Good	16.4%	16.2%	19.2%
	Regular	14.6%	15.1%	16.3%
	Unreliable	20.2%	25.2%	15.6%
	Reliability	**79.8%**	**74.8%**	**84.4%**
Patient Attendances	Very good	53.5%	50.0%	53.3%
	Good	13.0%	12.0%	12.4%
	Regular	15.7%	15.8%	15.0%
	Unreliable	17.8%	22.2%	18.3%
	Reliability	**82.2%**	**77.8%**	**80.7%**
Unallocated Patients with DTA	Very good	32.1%	28.7%	36.2%
	Good	15.7%	15.0%	16.0%
	Regular	10.5%	11.6%	11.1%
	Unreliable	41.7%	45.0%	36.7%
	Reliability	**58.3%**	**55.0%**	**63.3%**
Medically Fit for Discharge	Very good	53.5%	44.0%	53.4%
	Good	19.3%	17.4%	22.3%
	Regular	10.7%	14.6%	13.1%
	Unreliable	16.4%	24.1%	10.2%
	Reliability	**83.5%**	**76.0%**	**88.8%**

In addition, the average RMSE is also provided (Table 3) for each model and indicator, as the most neutral statistical residual measurement, even though, for the reasons outlined above, it may not the best fit for the ED environment.

Table 3. Accuracy comparison of ARIMA, Prophet and GRNN.

RMSE (Root Mean Square Deviation)			
	ARIMA	**Prophet**	**GRNN**
Patients in Department	18.1091	21.5031	17.7061
Patient Attendances	7.1002	8.8217	7.5965
Unallocated Patients with DTA	6.6968	8.1521	6.2153
Medically Fit for Discharge	35.1270	39.2894	34.3455

In both cases, green highlighting indicates the model with the best results.

8.2. Discussion

As can be seen, and as already previewed above, the machine learning technique GRNN usually provides the best results for this dataset and usually outperforming ARIMA, which is traditionally used for predicting ED indicators. However, ARIMA usually comes a close second and actually outperforms GRNN for the Patient Attendances, probably due to the very high variance in this indicator.

The relative performance is heavily impacted by the ability of GRNN to adapt rapidly to sudden dramatic changes in the data trends and for more stable time-series data, the performance of all three is likely to be closer.

The COVID-19 pandemic drove just such a change and strongly altered the behaviour of ED indicators, which then presented a challenge to the purely autoregressive models represented by ARIMA and Prophet, which rely solely on abstractions of previous values. GRNN was able to adapt to the change in data patterns promptly, resulting in closer predictions during this change period for all 4 indicators, as seen in Section 7. In addition, when analysing the entire 10 months of data for reliability (Table 2) and accuracy (Table 3), GRNN was the best fit model for 3 out of the 4 indicators, as seen in Section 8.

Although the difference in both accuracy and reliability are evident and relevant, data analysis suggests that outside of the pandemic period, the improvements in accuracy and reliability are reduced to slim margins. Hence, further simulations with different KPIs and data sample periods are necessary to further validate the findings of this study.

9. Conclusions

The main conclusion of this paper is that prediction of Emergency Department (ED) indicators can be improved by using machine learning models such as GRNN in comparison to traditional models such as ARIMA, particularly when sudden changes are observed in the data.

Indeed, it may be the case that ARIMA is better overall when indicators are very stable and so the proposed extension of this study is aimed at creating hybrid predictions, mixing observed predictions of ARIMA, GRNN, and possibly even Prophet, into a mixed time-series forecasting model, based on the short-term seasonal accuracy factors achieved by each model.

Author Contributions: Conceptualization, all authors; methodology, D.D.; software, D.D.; investigation, D.D.; data curation, D.D.; writing—original draft preparation, D.D.; writing—review and editing, C.W. and N.R.; visualization, D.D.; supervision, C.W. and N.R.; funding acquisition, C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by an Innovate UK Knowledge Transfer Partnership, KTP10507.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from Transforming Systems and are available from the D.D. subject to the permission of Transforming Systems.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Luo, L.; Luo, L.; Zhang, X.; He, X. Hospital daily outpatient visits forecasting using a combinatorial model based on ARIMA and SES models. *BMC Health Serv. Res.* **2017**, *17*, 469. [CrossRef] [PubMed]
2. Permanasari, E.; Hidayah, I.; Bustoni, I.A. SARIMA (Seasonal ARIMA) implementation on time series to forecast the number of Malaria incidence. In Proceedings of the 2013 International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, 7–8 October 2013; pp. 203–207. [CrossRef]
3. Sun, Y.; Heng, B.H.; Seow, Y.T.; Seow, E. Forecasting daily attendances at an emergency department to aid resource planning. *BMC Emerg. Med.* **2009**, *9*, 1. [CrossRef] [PubMed]
4. Jones, S.S.; Evans, R.S.; Allen, T.L.; Thomas, A.; Haug, P.J.; Welch, S.J.; Snow, G.L. A multivariate time series approach to modeling and forecasting demand in the emergency department. *J. Biomed. Inform.* **2009**, *42*, 123–139. [CrossRef] [PubMed]
5. Kunst, R.; Neusser, K. A forecasting comparison of some var techniques. *Int. J. Forecast.* **1986**, *2*, 447–456. [CrossRef]
6. Aboagye-Sarfo, P.; Mai, Q.; Sanfilippo, F.M.; Preen, D.B.; Stewart, L.M.; Fatovich, D.M. A comparison of multivariate and univariate time series approaches to modelling and forecasting emergency department demand in Western Australia. *J. Biomed. Inform.* **2015**, *57*, 62–73. [CrossRef] [PubMed]
7. Zhu, T.; Luo, L.; Zhang, X.; Shi, Y.; Shen, W. Time-Series Approaches for Forecasting the Number of Hospital Daily Discharged Inpatients. *IEEE J. Biomed. Health Inform.* **2017**, *21*, 515–526. [CrossRef] [PubMed]
8. Afilal, M.; Yalaoui, F.; Dugardin, F.; Amodeo, L.; Laplanche, D.; Blua, P. Forecasting the Emergency Department Patients Flow. *J. Med. Syst.* **2016**, *40*, 175. [CrossRef] [PubMed]
9. Milner, P.C. Ten-year follow-up of arima forecasts of attendances at accident and emergency departments in the Trent region. *Stat. Med.* **1997**, *16*, 2117–2125. [CrossRef]

10. Taylor, S.J.; Letham, B. Forecasting at Scale. *PeerJ Prepr.* **2017**, *5*, 25. [CrossRef]
11. Bani-Hani, D.; Khasawneh, M. A Recursive General Regression Neural Network (R-GRNN) Oracle for classification problems. *Expert Syst. Appl.* **2019**, *135*, 273–286. [CrossRef]
12. Apaydin, H.; Feizi, H.; Sattari, M.T.; Colak, M.S.; Shamshirband, S.; Chau, K.W. Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting. *Water* **2020**, *12*, 1500. [CrossRef]
13. Xu, Q.; Tsui, K.-L.; Jiang, W.; Guo, H. A Hybrid Approach for Forecasting Patient Visits in Emergency Department. *Qual. Reliab. Engng. Int.* **2016**, *32*, 2751–2759. [CrossRef]
14. Channouf, N.; L'Ecuyer, P.; Ingolfsson, A.; Avramidis, A.N. The application of forecasting techniques to modeling emergency medical system calls in Calgary, Alberta. *Health Care Manag. Sci.* **2007**, *10*, 25–45. [CrossRef] [PubMed]
15. Box, G.E.P.; Jenkins, G.M. *Time Series Analysis: Forecasting and Control*, 2nd ed.; University of Michigan, Holden-Day: Ann Arbor, MI, USA, 1976.
16. Vandaele, W. *Applied Time Series and Box-Jenkins Models*, 2nd ed.; Academic Press, Inc.: San Diego, CA, USA, 1983.
17. McCoy, T.H.; Pellegrini, A.M.; Perlis, R.H. Assessment of Time-Series Machine Learning Methods for Forecasting Hospital Discharge Volume. *JAMA Netw. Open* **2018**, *1*, e184087. [CrossRef] [PubMed]
18. Specht, D.F. *Probabilistic Neural Networks and General Regression Neural Networks*; Palo Alto: Santa Clara, CA, USA, 1995; Volume 67.
19. Al-Mahasneh, J.; Anavatti, S.G.; Garratt, M.A.; Pratama, M. Evolving General Regression Neural Networks Using Limited Incremental Evolution for Data-Driven Modeling of Non-Linear Dynamic Systems. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, Bangalore, India, 18–21 November 2018; pp. 335–341. [CrossRef]
20. Fisher, E.; O'dowd, N.C.; Dorning, H.; Keeble, E.; Kossarova, L. Quality at a Cost. 2016. Available online: http://www.qualitywatch.org.uk/sites/files/qualitywatch/field/field_document/QWannualstatement2016%28final%29WEB.pdf (accessed on 19 February 2018).
21. Stephens, M.; Cross, S.; Luckwell, G. Coronavirus and the Impact on Output in the UK Economy. Office for National Statistics. 2020; pp. 1–29. Available online: https://www.ons.gov.uk/economy/grossdomesticproductgdp/articles/coronavirusandtheimpactonoutputintheukeconomy/april2020 (accessed on 31 October 2020).
22. Thorlby, R.; Tinson, A.; Kraindler, J. COVID-19: Five Dimensions of Impact | The Health Foundation. No. April. 2020, pp. 1–20. Available online: https://www.health.org.uk/news-and-comment/blogs/covid-19-five-dimensions-of-impact (accessed on 31 October 2020).
23. Krollner, B.; Vanstone, B.; Finnie, G. Financial time series forecasting with machine learning techniques: A survey. In Proceedings of the 8th European Symposium on Artificial Neural Networks, ESANN 2010, Bruges, Belgium, 28–30 April 2010; pp. 25–30.

applied sciences

MDPI

Article

Data-Driven Real-Time Online Taxi-Hailing Demand Forecasting Based on Machine Learning Method

Zhizhen Liu, Hong Chen *, Xiaoke Sun and Hengrui Chen

College of Transportation Engineering, Chang'an University, Xi'an 710000, China; 2018021077@chd.edu.cn (Z.L.); 2019021061@chd.edu.cn (X.S.); 2019021060@chd.edu.cn (H.C.)

* Correspondence: glch@chd.edu.cn; Tel.: +86-137-0022-9619

Received: 13 August 2020; Accepted: 22 September 2020; Published: 24 September 2020

Featured Application: This research provides a valuable data-driven method on forecasting the online taxi-hailing demand, and it could be potentially applied to developing multi-modes transportation prediction.

Abstract: The development of the intelligent transport system has created conditions for solving the supply–demand imbalance of public transportation services. For example, forecasting the demand for online taxi-hailing could help to rebalance the resource of taxis. In this research, we introduced a method to forecast real-time online taxi-hailing demand. First, we analyze the relation between taxi demand and online taxi-hailing demand. Next, we propose six models containing different information based on backpropagation neural network (BPNN) and extreme gradient boosting (XGB) to forecast online taxi-hailing demand. Finally, we present a real-time online taxi-hailing demand forecasting model considering the projected taxi demand ("PTX"). The results indicate that including more information leads to better prediction performance, and the results show that including the information of projected taxi demand leads to a reduction of MAPE from 0.190 to 0.183 and an RMSE reduction from 23.921 to 21.050, and it increases R^2 from 0.845 to 0.853. The analysis indicates the demand regularity of online taxi-hailing and taxi, and the experiment realizes real-time prediction of online taxi-hailing by considering the projected taxi demand. The proposed method can help to schedule online taxi-hailing resources in advance.

Keywords: online taxi-hailing demand; backpropagation neural network; extreme gradient boosting; real-time prediction

1. Introduction

With the development of the intelligent transportation system, the travel of residents is growing more convenient. Nevertheless, because of the information asymmetry between passengers and drivers, the spatial and temporal distribution of passengers and drivers are inconsistent. The limited urban transportation resources were wasted by the information asymmetry between passengers and drivers. Therefore, trip demand in the urban area urgently needs to be studied. Recently, online taxi-hailing has gradually become the primary trip mode for urban residents. Meanwhile, the taxi still assumes the function of public transportation for urban residents. Under these circumstances, the online taxi-hailing demand would be affected by the taxi demand because of the homogeneity between the taxi and online taxi-hailing. Thus, we should take the taxi demand into account while studying the online taxi-hailing demand.

In the past, research that focused on forecasting traffic demand was mostly based on environmental data and GPS data [1–32]. Moreover, the research mined the features of GPS data and environmental data to forecast the trip demand, while the research ignored the relationship between the taxi and online taxi-hailing.

Therefore, this study aims to enhance the prediction effects of forecasting online taxi-hailing demand considering the taxi demand. Moreover, this research is a follow-up experiment of [32]. First, we use Pearson correlation analysis to screen the determinative influence factors to enhance the prediction accuracy. Then, online taxi-hailing demand forecasting models based on extreme gradient boosting (XGB) and backpropagation neural network (BPNN) were introduced to explore the relationship between taxi demand and online taxi-hailing demand. Next, we realize the real-time forecasting of online taxi-hailing demand by proposing a data-driven prediction method. This study would help to enhance the accuracy of online taxi-hailing demand forecasting and is essential for rebalancing traffic resources.

The literature review related to our study is presented in Section 2. Section 3 describes the data and the preprocessing of data in this study. Next, we proposed methods to enhance the accuracy of predicting online taxi-hailing demand in Section 4, while Section 5 concludes the results. Finally, the discussion and conclusion are shown in Section 6.

2. Related Work

Over the years, numerous works have been dedicated to enhancing the accuracy of trip demand forecasting. The first application of the trip demand forecasting is predicting trip demand based on a four-step process considering spatiotemporal factors [1]. L. Moreira-Matias et al. predicted the spatial distribution of taxi demand by presenting a method [2]. Then, he proposed a learning model considering real-time data to forecast the taxi-passenger demand's spatiotemporal distribution [3]. Next, he proposed a combination forecasting model to forecast the taxi-passenger demand's spatiotemporal distribution [4]. K. Zhang et al. forecasted the location of hotspots and tested the heat of the hotspots by presenting an adaptive forecasting method [5]. Next, N. Davis et al. proposed a time-series method to forecast the taxi demand by mining the regulation of taxi mobile app data [6]. X. Peng et al. forecasted the taxi demand hotspots based on social media check-ins to reduce the imbalanced supply and demand of taxis [7]. K. Zhao et al. predicted the taxi demand through three forecasting methods, respectively, based on the Markov model, Lempel–Ziv–Welch model, and ANN model [8]. Besides the GPS data and environmental data, J. Xu et al. also considered historical traffic behaviors as an important variable in the taxi demand forecasting problem, and they proposed an LSTM method to forecast taxi demand in several urban areas [9]. D. Zhang improved the hidden Markov chain model and proposed a D-model to forecast the taxi demand [10]. For exploring the relationship between taxi and subway, Y. Bao et al. took the interaction between taxi demand and subway demand into account to explore the impacts of the interaction on the accuracy of taxi demand and proposed a taxi demand prediction method based on a neural network model [11]. N. Davis explored the impacts of tessellation on-demand prediction effects and proposed a combination algorithm of different tessellation strategies to predict taxi demand [12].

The research above considered the impacts of the GPS data and the environmental data on prediction accuracy, but they did not take real-world event information into account. To address this problem, I. Markou et al. mined the real-world event information from unstructured data, and they applied the machine learning method to realize taxi demand forecasting [13]. S. Ishiguro et al. introduced the real-time demographic data into the taxi demand forecasting method and explored the impacts of demographic data on taxi demand forecasting accuracy by a stacked denoising autoencoder [14]. S. Liao conducted a comparison of two deep neural networks for forecasting trip demand and found that DNNs perform better than other traditional machine learning methods [15]. U. Vanichrujee et al. presented an ensemble method consisting of the LSTM model, GRU model, and extreme gradient boosting model (XGB) to forecast taxi demand [16]. J. Xu proposed a sequence learning method considering the historical demand to forecast trip demand [17]. H. Yao et al. presented a multi-view spatiotemporal network framework to simulate spatiotemporal relationships and forecasted the traffic demand [18]. H. Yan analyzed taxi requests and proposed a Bayesian hierarchical semiparametric model to forecast taxi demand [20]. L. Kuang introduced the unstructured data into a deep learning method to forecast the trip demand [21]. However, the methods above ignored the destination of passengers. L. Liu proposed a method to forecast the taxi demand between origin–destination pairs [22]. I. Markou introduced real-world events into the prediction method and used the data to forecast traffic demand [23]. F. Rodrigues et al. explored the relationship between drop-off points and pick-up points and proposed a spatio-temporal LSTM model to forecast the taxi demand [25]. F. Terroso-Saenz predicted taxi demand through the QUADRIVEN method based on human-generated data [26]. Y. Xu proposed a graph and time-series learning model considering the relationships between non-adjacent for city-wide taxi demand prediction [27]. H. Yu proposed a deep spatiotemporal recurrent convolutional neural network to forecast traffic flow [28]. X. Liu explored the impacts of the socio-economic, transport system, and land-use patterns on taxi demand forecasting [29]. A. Saadallah introduced the BRIGHT method, which is an ensemble of time series analysis models to forecast taxi demand precisely [30]. A. Safikhani proposed a STAR model to analyze the spatiotemporal distribution of taxis and introduced the LASSO-type penalized methods to tackle parameter estimation [31]. Recently, Z. Liu proposed a combination forecasting model considering the random forest method and ridge regression method to predict taxi demand in hotspots [32].

In general, given the relationship between different trip modes, more attempts can be justified. This study is initiated by a real-world case study to better understand the underlying relationship between the demands of different trip modes.

3. Data Description

3.1. Taxi GPS Data

We obtained the GPS data from the Xi'an Taxi Management Office in Xi'an city of China. The data include location information, vehicle state information, time information, and license plate information. Moreover, the taxi GPS data were recorded every 5 s for 30 days in November 2016 and include 40 million points which are located in Xi'an city of China. The GPS data were cleaned and selected. An instance of taxi GPS data is shown in Table 1.

Table 1. An instance of taxi GPS data.

Field	Type	Sample	Comment
License plate number	String	BMvCh8nxqktd xniovIFuns	Anonymized
Time	String	1 November 2016 00:01:35	Time
Longitude	String	108.908109	WGS84 Coordinate System
Latitude	String	34.235744	WGS84 Coordinate System
Vehicle state	String	1	0: Without passenger 1: With passenger

3.2. Online Taxi-Hailing GPS Data

Online taxi-hailing GPS data are from Didi Chuxing GAIA Initiative, and the GPS data are located in Xi'an city of China. The dataset consists of 600 million track points, and it was recorded every 2–4 s for 30 days in November 2016. An instance of online taxi-hailing GPS data is shown in Table 2.

Table 2. An instance of online taxi-hailing GPS data.

Field	Type	Sample	Comment
Driver ID	String	glox.jrrlltBMvCh8n xqktdr2dtopmlH	Anonymized
Order ID	String	jkkt8kxniovIFuns9q rrlvst@iqnpkwz	Anonymized
Time Stamp	String	1478223613 (4 November 2016 09:40:13)	Unix timestamp, in seconds
Longitude	String	104.04392	GCJ-02 Coordinate System
Latitude	String	34.04392	GCJ-02 Coordinate System

3.3. Environmental Data

The environmental data conclude air quality data and meteorological data. The air quality data in Xi'an city are from the official website of Green Breathing. The meteorological data in Xi'an city were derived from the National Meteorological Information Center. This study selects the hourly environmental data of Xi'an. In general, the environmental data contain 15 dimensions for the research (Table 3).

Table 3. Environmental data structure description.

Data	Indicators	Description
Air quality data	AQI	Air Quality Index
	CO	The concentration of CO ($\mu g/m^3$)
	NO_2	The concentration of NO_2 ($\mu g/m^3$)
	O_3	The concentration of O_3 ($\mu g/m^3$)
	PM2.5	The concentration of PM2.5 ($\mu g/m^3$)
	PM10	The concentration of PM10 ($\mu g/m^3$)
	SO_2	The concentration of SO_2 ($\mu g/m^3$)
Meteorological data	W	Weather. 1: Sunny 2: Cloudy 3: Raining 4: Haze
	WS	Wind speed (m/s)
	TEM	Temperature (°C)
	SSD	Sunshine duration (h)
	PRE	Precipitation (mm)
	TG	Sensible temperature (°C)
	VIS	Horizontal visibility (m)

4. Methods

4.1. Feature Selection

Ensuring that the correlations between the features and the dependent variables are important in the prediction problem. Likewise, ensuring that the features are independent of one another is also important for improving the prediction accuracy. While modeling the forecasting method, both the features which exhibit strong, multiple collinearities and the features which have a low correlation with the dependent variable should be eliminated for enhancing the prediction accuracy. Thus, we choose the Pearson correlation analysis to test the correlation of all features and the dependent variable [33,34]. The calculation of Pearson correlation analysis is as Equation (1).

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_X)]}{\sigma_X \sigma_Y}, \tag{1}$$

$\mathrm{cov}(X,Y)$ is the covariance between the features X and Y. σ_X and σ_Y indicate the standard deviations of the features X and Y. $\rho_{X,Y}$ is the correlation value of the features X and Y. The value range of $\rho_{X,Y}$ belongs to $(-1,1)$. If $\rho_{X,Y} > 0$, the two features are positively correlated: if $\rho_{X,Y} < 0$, the two features are negatively correlated. The larger absolute value of $\rho_{X,Y}$ indicates a stronger correlation between the features X and Y.

4.2. BPNN

Artificial neural networks (ANNs) possess attributes of learning, generalizing, parallel processing, and error endurance. These attributes make the ANNs useful in modeling complex situations. Therefore, we employ BPNN, a type of ANN, for forecasting online taxi-hailing demand in this study [35,36]. A three-layer BPNN employed in this paper is shown in Figure 1 [37]. In Figure 1, "T" indicates the information of time factors, "E" is the information of environmental factors, and "TX" represents the information of taxi demand.

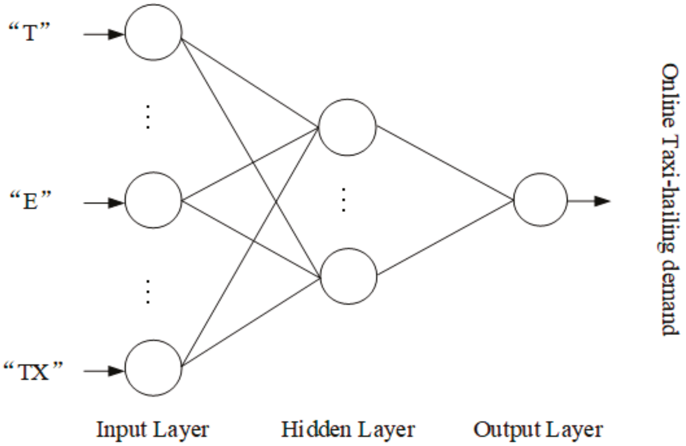


Figure 1. An instance of three-layer backpropagation neural network (BPNN).

The connection weights among nodes are obtained by data training in the backpropagation process. Then, it produces the minimized least-mean-square error between the true and the estimated values from the neural network's output. First, the connection weights are assigned initial values. Then, the weights are updated based on the back-propagated error between the predicted and true output values. Assume that there are n input neurons, m hidden neurons, and one output neuron, a training process can be described as follows.

Hidden layer stage: Calculating the outputs of all neurons in the hidden layer as Equations (2) and (3).

$$\mathrm{net_j} = \sum\nolimits_{i=0}^{n} v_{ij}x_i,\ j = 1, 2, \cdots, m, \tag{2}$$

$$y_j = f_H\big(\mathrm{net_j}\big),\ j = 1, 2, \cdots, m, \tag{3}$$

$\mathrm{net_j}$ is the activation value of the jth node, y_j is the output of the hidden layer, and f_H is the activation function of a node; the activation function is the rectified linear unit function as Equation (4).

$$f_H(x) = \max(0, x), \tag{4}$$

Output stage: The outputs of all neurons in the output layer are as Equation (5).

$$O = f_o\Big(\sum\nolimits_{j=0}^{m} w_{ik}y_j\Big) \tag{5}$$

f_o is the activation function as Equation (4). All weights are assigned random values initially and then modified by the delta rule according to the learning samples.

The three-layer BPNN above is the basic application of BPNN in online taxi-hailing demand prediction method. To find out the best network structure of BPNN for different forecasting models, we should use the grid search algorithm to determine the network structures of the models based on BPNN.

4.3. XGB

XGB is a boosting model based on a classification and regression tree (CART), which takes full advantage of the residual of a base classifier [38]. The boosting algorithm combines simple tree models to establish a more precise model, and it overcomes the influence of the interference signal. The prediction is as Equation (6).

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{6}$$

f_k is the k_{th} tree, K is the number of trees, and F is a set of all trees.

Suppose that $S = \{(x_1, y_1), \ldots (x_i, y_i) \ldots (X_w,\ y_w)\}$ is a known dataset with N samples where x has L features, and y is the label of different emitters. The objective function is as Equation (7).

$$O = \sum_{i=1}^{N} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} r(f_k), \tag{7}$$

$\hat{y}_i$ is the predicted value of x_i, l represents the difference between the true and predicted values. $r(f_k)$ is the regularized term of k_{th} trees, which penalize the complexity of the model to avoid overfitting, and it could be calculated as Equation (8).

$$r(f_k) = \gamma T + \frac{\omega}{2}\|\vartheta\|, \tag{8}$$

γ, ω are penalty coefficients, T is the number of leaves in the tree, and ϑ is leaf weight.

4.4. Evaluation Criteria

Moreover, three accuracy measures are applied to evaluate the performance of online taxi-hailing prediction. The measures are root-mean-square error (RMSE), mean absolute percentage error (MAPE) and goodness of fit (R^2), which are calculated as Equations (9)–(11).

$$RMSE = \left(T^{-1} \sum_{n=1}^{T} \left(\hat{C_n} - C_n\right)^2\right)^{1/2}, \tag{9}$$

$$MAPE = T^{-1} \sum_{n=1}^{T} \left|\left(\hat{C_n} - C_n\right)/\hat{C_n}\right|, \tag{10}$$

$$R^2 = \frac{\sum_{i=1}^{N} \left(C_n - \overline{C}\right)^2}{\sum_{i=1}^{N} \left(\hat{C_n} - \overline{C}\right)^2}, \tag{11}$$

$\hat{C_n}$, C_n and $\overline{C}$ are the true, the predicted, and the mean value, respectively. Then, T is the number of samples.

5. Results

5.1. Feature Selection

Before we predict the online taxi-hailing demand, we should select a reasonable set of forecasting features. Therefore, we use Python to calculate the correlations among prediction indicators, and we eliminate factors with strong collinearity and factors with low cross-correlation. Correlations among environmental factors are as Table 4. In Table 4, "OT" and "TX", respectively, indicate online taxi-hailing demand and taxi demand, “DW” is the day of the week, “HD” represents the hour of the day, and “WON” indicates whether the day is a workday. Other features are as Table 3.

Table 4. Correlations among environmental factors.

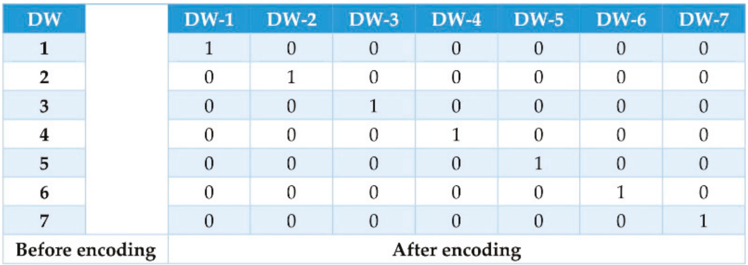
	OT	**DW**	**HD**	**WON**	**AQI**	**PM2.5**	**PM10**	**SO_2**	**NO_2**	**CO**	**O_3**	**W**	**WS**	**TEM**	**RHU**	**PRE**	**SSD**	**TG**	**VIS**	**TX**
OT	1	0.23	0.79	0.22	0.06	0.06	0.07	0.02	0.08	0.08	−0.18	−0.02	0.11	−0.39	0.41	0.02	0.06	−0.23	−0.27	0.52
DW	0.23	1	0	0.79	0.29	0.28	0.34	0.13	0.36	0.4	−0.25	−0.17	−0.04	−0.07	0.13	−0.02	0.22	−0.03	−0.18	0.06
HD	0.79	0	1	0.11	0.05	0.07	0.09	0.05	0.14	0.12	0.07	0	0.1	−0.38	0.34	0	0	−0.13	−0.17	0.5
WON	0.22	0.79	0.11	1	0.13	0.2	0.25	0.19	0.25	0.35	−0.15	−0.2	0.02	−0.06	0.19	−0.02	0.18	−0.04	−0.28	0.01
AQI	0.06	0.29	0.05	0.13	1	0.79	0.95	0.3	0.69	0.8	−0.35	0.31	−0.02	0.01	0.08	−0.04	−0.09	−0.09	−0.04	0.01
PM2.5	0.06	0.28	0.07	0.2	0.79	1	0.7	0.46	0.8	0.88	−0.53	0.51	0.13	0.02	0.15	−0.03	−0.29	−0.14	−0.29	0.02
PM10	0.07	0.34	0.09	0.25	0.95	0.7	1	0.37	0.67	0.8	−0.35	0.16	−0.01	−0.06	0.11	−0.05	0.07	−0.14	−0.06	0
SO_2	0.02	0.13	0.05	0.19	0.3	0.46	0.37	1	0.46	0.5	−0.34	0.15	0.48	−0.36	−0.11	−0.05	0	−0.56	−0.24	−0.04
NO_2	0.08	0.36	0.14	0.25	0.69	0.8	0.67	0.46	1	0.91	−0.36	0.23	0.04	0	0.18	−0.06	0.19	−0.11	−0.35	0.04
CO	0.08	0.4	0.12	0.35	0.8	0.88	0.8	0.5	0.91	1	−0.47	0.3	0	0.04	0.2	−0.03	0.03	−0.07	−0.34	0.02
O_3	−0.18	−0.25	0.07	−0.15	−0.35	−0.53	−0.35	−0.34	−0.36	−0.47	1	−0.02	0.02	−0.06	−0.13	−0.08	0.27	−0.02	0.17	−0.02
W	−0.02	−0.17	0	−0.2	0.31	0.51	0.16	0.15	0.23	0.3	−0.02	1	0.06	0.04	0.08	−0.03	−0.51	−0.07	−0.19	−0.05
WS	0.11	−0.04	0.1	0.02	−0.02	0.13	−0.01	0.48	0.04	0	0.02	0.06	1	−0.53	−0.2	−0.07	−0.14	−0.68	−0.02	0.01
TEM	−0.39	−0.07	−0.38	−0.06	0.01	0.02	−0.06	−0.36	0	0.04	−0.06	0.04	−0.53	1	−0.22	0.03	−0.08	0.8	0.13	−0.2
RHU	0.41	0.13	0.34	0.19	0.08	0.15	0.11	−0.11	0.18	0.2	−0.13	0.08	−0.2	−0.22	1	0.11	0.09	0.12	−0.72	0.27
PRE	0.02	−0.02	0	−0.02	−0.04	−0.03	−0.05	−0.05	−0.06	−0.03	−0.08	−0.03	−0.07	0.03	0.11	1	−0.03	0.07	−0.01	0.05
SSD	0.06	0.22	0	0.18	−0.09	−0.29	0.07	0	0.19	0.03	0.27	−0.51	−0.14	−0.08	0.09	−0.03	1	0.05	−0.05	0.03
TG	−0.23	−0.03	−0.13	−0.04	−0.09	−0.14	−0.14	−0.56	−0.11	−0.07	−0.02	−0.07	−0.68	0.8	0.12	0.07	0.05	1	0.01	−0.13
VIS	−0.27	−0.18	−0.17	−0.28	−0.04	−0.29	−0.06	−0.24	−0.35	−0.34	0.17	−0.19	−0.02	0.13	−0.72	−0.01	−0.05	0.01	1	−0.16
TX	0.52	0.06	0.5	0.01	0.01	0.02	0	−0.04	0.04	0.02	−0.02	−0.05	0.01	−0.2	0.27	0.05	0.03	−0.13	−0.16	1

As shown in Table 4, we find that the values of correlations among AQ, AQI, PM2.5, PM10, and CO are more than 0.8. Therefore, we remove AQI, PM2.5, PM10, and CO from the predictive factors. Next, we eliminate the features whose correlation with the OT factor is less than 0.2. Predictive indicators of online taxi-hailing demand areas are shown in Table 5. Predictive indicators in Table 5 are divided into “T”, “E” and “TX”. “T” indicates the information of the time, “E” represents the environmental factors, and “TX” contains information about taxi demand.

Table 5. Predictive indicators of trip demand.

Information	Indicators	Description
T	DW	Day of the week. DW is in (1, 7).
	HD	Hour of day. HD is in (1, 24).
	WON	Workday or non-workday. Workday: 0; Non-workday: 1.
E	WS	Wind speed (m/s).
	TEM	Temperature (°C).
	RHU	Relative humidity (%).
	TG	Sensible temperature (°C).
	VIS	Horizontal visibility (m).
TX	TX	Taxi demand (pcu).

Then, all data are proceeded through by the One-Hot Encoder function in the scikit-learn. preprocessing library. An instance of the DW indicator is shown in Figure 2.



DW		DW-1	DW-2	DW-3	DW-4	DW-5	DW-6	DW-7
1		1	0	0	0	0	0	0
2		0	1	0	0	0	0	0
3		0	0	1	0	0	0	0
4		0	0	0	1	0	0	0
5		0	0	0	0	1	0	0
6		0	0	0	0	0	1	0
7		0	0	0	0	0	0	1
Before encoding		After encoding						

Figure 2. An instance of One-Hot Encoder.

After the encoding of indicators in Table 5, the dimension of the dataset was expanded to 58. Additionally, the first 23 days of November 2016 are taken as the training set, with the other seven days as the testing set in this study.

5.2. Data Preprocessing

In this study, we choose the Bell Tower area as the research object according to the study of Liu et al. [32], because the Bell Tower area contains the most trip demand. The Bell Tower area is a commercial area, and its traffic demand exhibits a robust tidal phenomenon. The Bell Tower area is as in Figure 3.

Then, we cut taxi data and online taxi-hailing data into time slices. The trip demand for taxi and online taxi-hailing areas is shown in Figure 4. We find that the taxi demand and online taxi-hailing demand are regular, and taxi demand decreases while the online taxi-hailing demand increases in peak hours.

5.3. Online Taxi-Hailing Demand Forecasting

Then, we forecast the online taxi-hailing demand in Bell Tower area based on the BPNN and XGB. We test the prediction effects of different indicators based on the BPNN and XGB. In the experiment, we add time factors, environmental factors, and taxi demand factors into models based on BPNN and XGB. Models with different impacting factors are shown in Table 6. Next, we use the grid search algorithm to adjust the hyperparameters of models based on BPNN and XGB. Moreover, the hyperparameters for the models are illustrated in Table 7. Furthermore, the results of models with different impacting factors are shown in Figures 5 and 6. Additionally, the factors of "T", "E" and "TX" are shown in Table 5.

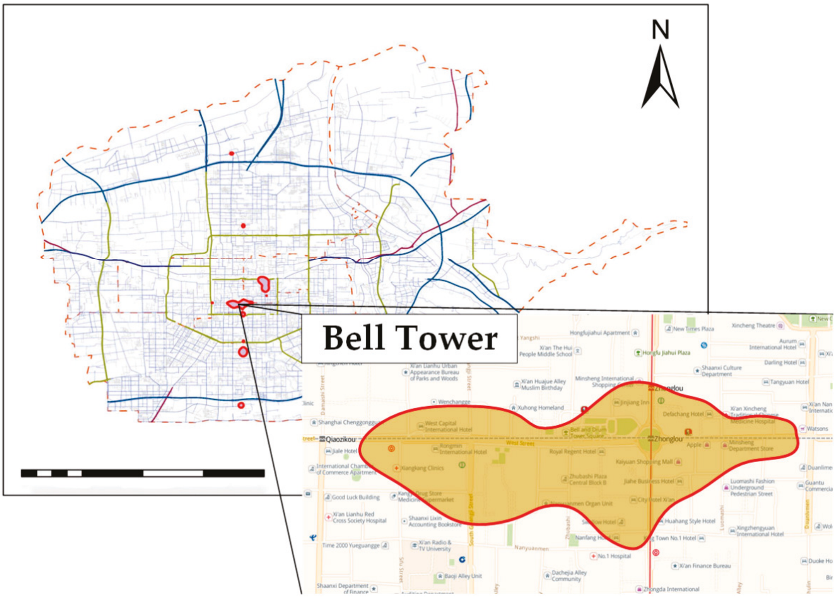


Figure 3. Research scope in the Bell Tower area.

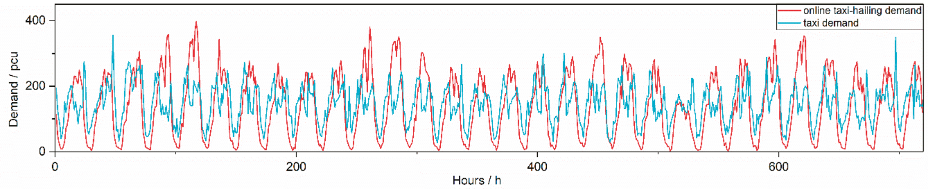


Figure 4. The demand for taxi and online taxi-hailing in Bell Tower area.

Table 6. Predictive indicators of trip demand.

Model	Model Input	Model Output
Model BPNN + T	Time data	Next week's demand
Model BPNN + T + E	Time data + Environmental data	Next week's demand
Model BPNN + T + TX	Time data + Environmental data + Taxi data	Next week's demand
Model XGB + T	Time data	Next week's demand
Model XGB + T + E	Time data + Environmental data	Next week's demand
Model XGB + T + TX	Time data + Environmental data + Taxi data	Next week's demand

Table 7. Hyperparameter settings for the models.

Model	Hyperparameters	Value	Optimal Value
Model BPNN + T	m (hidden neurons)	{1, 2, . . . , 100}	67
	n (hidden layers)	{1, 2, 3, 4}	3
	activation	{'identity', 'logistic', 'tanh', 'relu'}	'relu'
Model BPNN + T + E	m (hidden neurons)	{1, 2, . . . , 100}	89
	n (hidden layers)	{1, 2, 3, 4}	3
	activation	{'identity', 'logistic', 'tanh', 'relu'}	'relu'

Table 7. *Cont.*

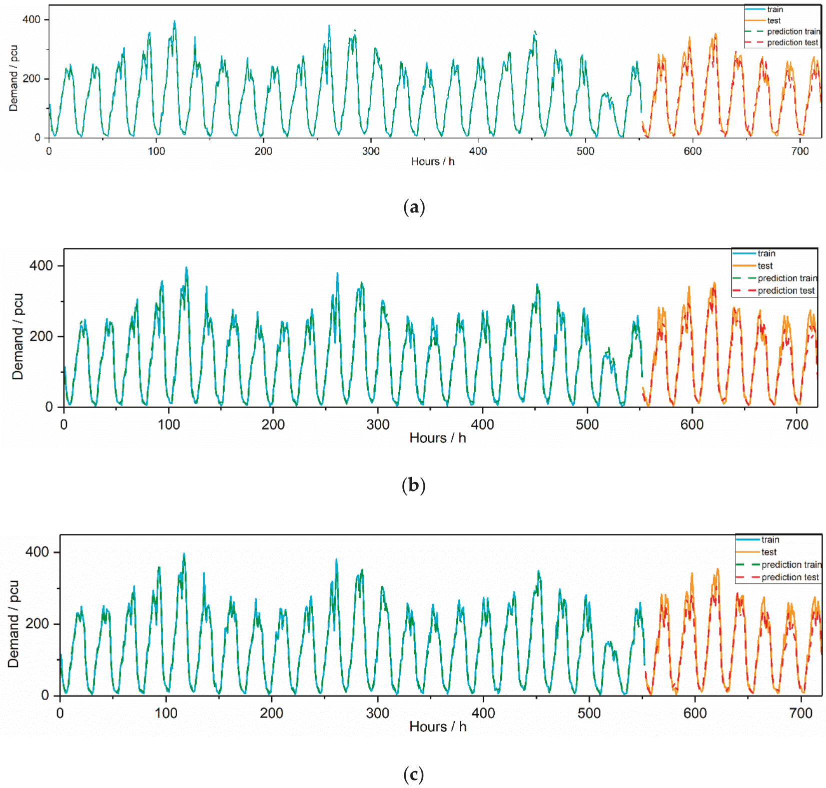
Model	Hyperparameters	Value	Optimal Value

Model BPNN + T + TX	m (hidden neurons)
	n (hidden layers)
	activation
		{1, 2, . . . , 100}
		{1, 2, 3, 4}
		{'identity', 'logistic', 'tanh', 'relu'}
			88
			3
			'relu'

Model XGB + T	gamma
	learning_rate
	max_depth
	min_child_weight
	n_estimators
		{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
		{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
		{1, 2, . . . , 50}
		{1, 2, . . . , 50}
		{1, 2, . . . , 100}
			0.4
			0.4
			13
			12
			48

Model XGB + T + E	gamma
	learning_rate
	max_depth
	min_child_weight
	n_estimators
		{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
		{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
		{1, 2, . . . , 50}
		{1, 2, . . . , 50}
		{1, 2, . . . , 100}
			0.4
			0.3
			11
			12
			38

Model XGB + T + TX	gamma
	learning_rate
	max_depth
	min_child_weight
	n_estimators
		{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
		{0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
		{1, 2, . . . , 50}
		{1, 2, . . . , 50}
		{1, 2, . . . , 100}
			0.4
			0.3
			11
			11
			45

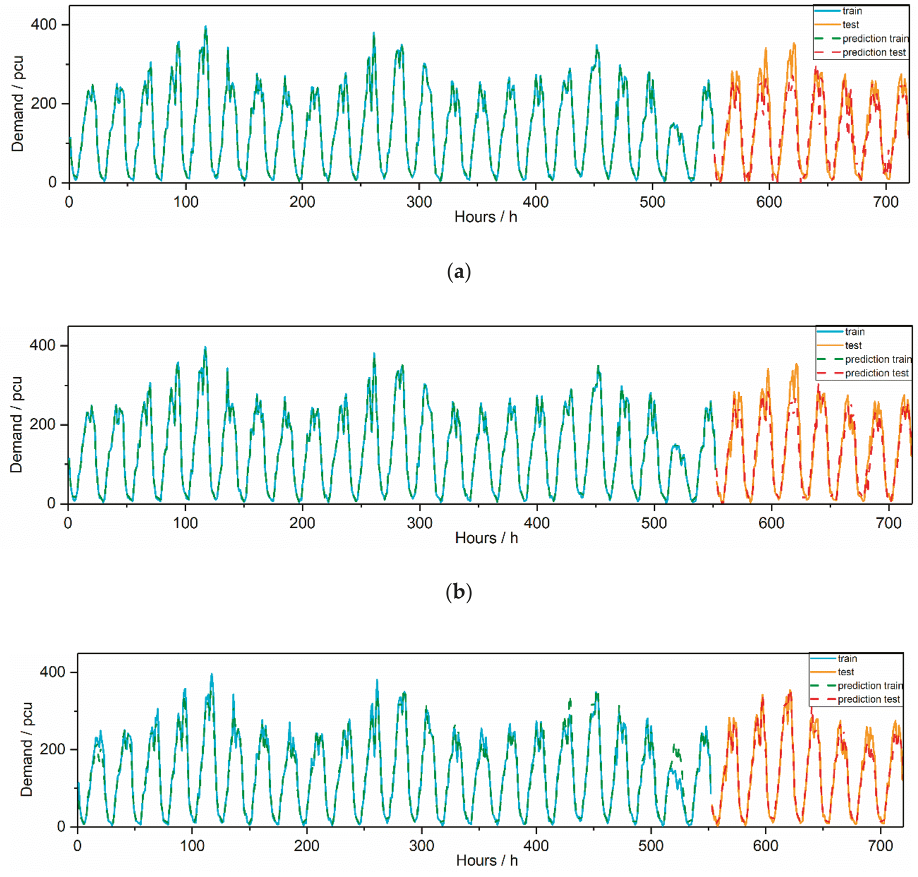


(**a**)

(**b**)

(**c**)

Figure 5. Online taxi-hailing demand prediction results of models based on BPNN. (**a**) Result of model "BPNN + T"; (**b**) Result of model "BPNN + T + E"; (**c**) Result of model "BPNN + T + E + TX".



(**a**)

(**b**)

(**c**)

Figure 6. Online taxi-hailing demand prediction results of models based on extreme gradient boosting (XGB). (**a**) Result of model "BPNN + T"; (**b**) Result of model "BPNN + T + E"; (**c**) Result of model "BPNN + T + E + TX".

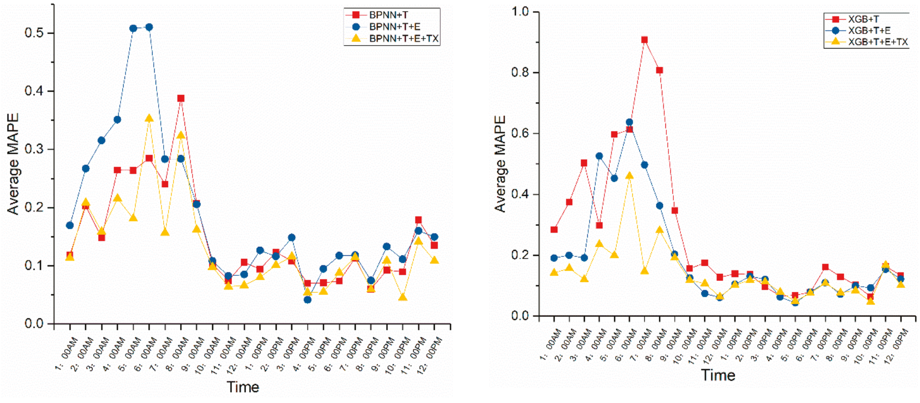
Then we use RMSE, MAPE and R^2 to test the prediction effect of the models (Table 8). Table 8 shows the RMSE, MAPE and R^2 of six different models' test datasets in the Bell Tower area. Comparing the performance of predictions based on BPNN, our results show that the model "BPNN + T + E + TX" is the best-performing method for solving online taxi-hailing prediction problems. Moreover, among three predictions based on XGB, the model "XGB + T + E + TX" is the best-performing method for online taxi-hailing prediction problems.

Next, we analyze the contributions of the different sources of information. From Table 8, we can find that including information about taxi demand ("TX") enhances the prediction effects based on BPNN and XGB. In the BPNN models, including information "E" leads to a MAPE reduction from 0.224 to 0.190, while it decreases RMSE from 28.576 to 23.921, and increases the R^2 from 0.819 to 0.845. Likewise, including information "TX" leads to a MAPE reduction from 0.190 to 0.132, and it increases the R^2 from 0.845 to 0.866. Meanwhile, in the XGB models, including information "E" leads to a MAPE reduction from 0.333 to 0.197 while it reduces RMSE from 26.296 to 21.206, and increases the R^2 from 0.833 to 0.857. Including information "TX" leads to a MAPE reduction from 0.197 to 0.139, and it increases the R^2 from 0.857 to 0.865. Additionally, the performance of the model "BPNN + T + E + TX" is the best among the six models in Table 8.

Table 8. Prediction effects of BPNN and XGB.

Models	**Training Set**			**Testing Set**		
	MAPE	**RMSE**	**R^2**	**MAPE**	**RMSE**	**R^2**
BPNN + T	0.213	26.972	0.842	0.224	28.576	0.819
BPNN + T + E	0.178	23.692	0.851	0.190	23.921	0.845
BPNN + T + E + TX	0.127	18.632	0.893	0.132	18.696	0.866
XGB + T	0.302	24.701	0.836	0.333	26.296	0.833
XGB + T + E	0.188	19.718	0.872	0.197	21.206	0.857
XGB + T + E + TX	0.129	17.797	0.887	0.139	19.193	0.865

To evaluate the prediction performance of BPNN and XGB in different hours, we report MAPE and RMSE of six models in different hours. Figure 7a shows that the model "BPNN + T + E + TX" obtains the lowest MAPE among three predictions except at 6 a.m., 8 a.m., and 9 p.m. Figure 7b shows that the performance of the model "XGB + T + E + TX" is the best except at 11 a.m. and 5 p.m. From Figure 8, we know that the model "BPNN + T + E + TX" obtains the lowest RMSE among three predictions except at 4, 7, and 9 p.m., and performances of the model "XGB + T + E + TX" are the best except at 11 a.m., 12 a.m., 4 p.m. and 5 p.m.

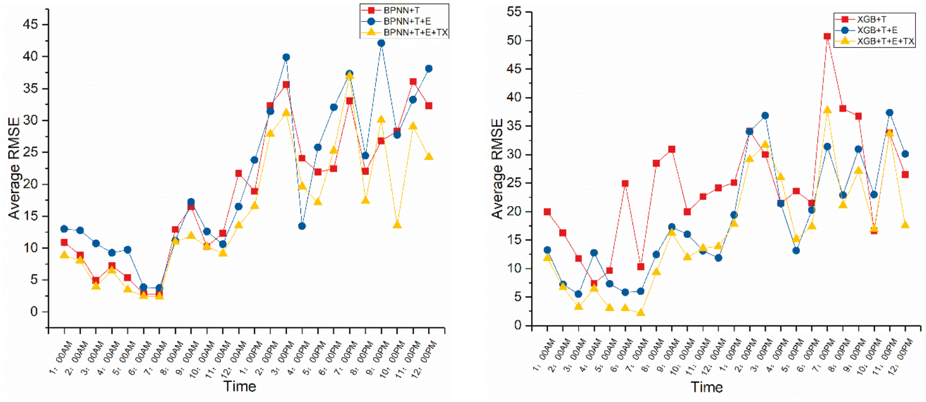


(**a**) (**b**)

Figure 7. Average mean absolute percentage error (MAPE) of online taxi-hailing demand prediction based on BPNN and XGB in different hours. (**a**) MAPE of the prediction models based on BPNN; (**b**) MAPE of the prediction models based on XGB.

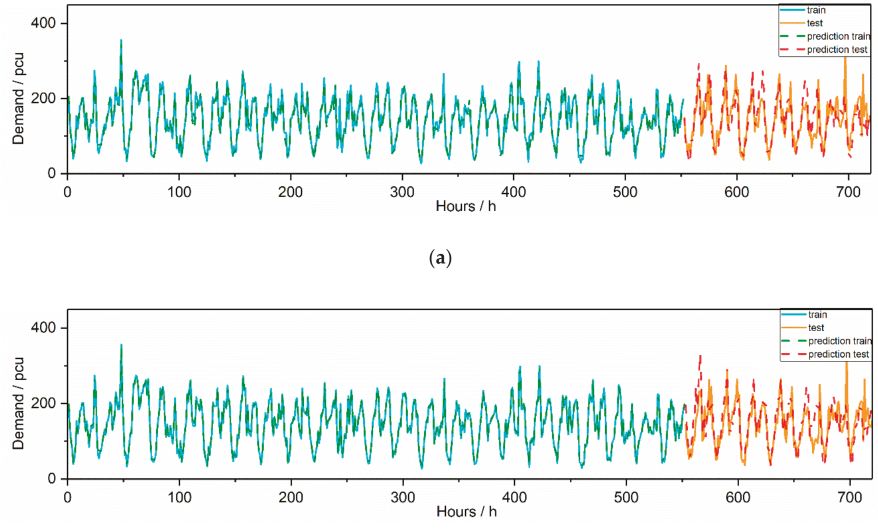
5.4. Real-Time Online Taxi-Hailing Demand

While we are forecasting online taxi-hailing demand by different models in Table 6, we ignore that the future taxi demand is unavailable. To realize the real-time online taxi-hailing demand prediction, we should predict the taxi demand before forecasting the online taxi-hailing demand by model "BPNN + T + E" and "XGB + T + E". The results of taxi demand prediction are as in Figure 9 and Table 9.



(**a**) (**b**)

Figure 8. Average RMSE of online taxi-hailing demand prediction based on BPNN and XGB in different hours. (**a**) RMSE of the prediction models based on BPNN; (**b**) RMSE of the prediction models based on XGB.



(**a**)

(**b**)

Figure 9. The taxi demand prediction based on "BPNN + T + E" and "XGB + T + E". (**a**) Result of taxi demand prediction based on "BPNN + T + E"; (**b**) Result of taxi demand prediction based on "XGB + T + E".

Table 9. Prediction effects of taxi demand prediction based on BPNN and XGB.

Models	Training Set			Testing Set		
	MAPE	RMSE	R^2	MAPE	RMSE	R^2
BPNN + T + E	0.177	31.812	0.874	0.185	32.860	0.851
XGB + T + E	0.204	37.625	0.747	0.213	37.754	0.713

Table 9 shows that the model "BPNN + T + E" performs better than model "XGB + T + E" in forecasting taxi demand. Based on the information on taxi demand prediction ("PTX"), we forecast the online taxi-hailing demand by model "BPNN + T + E + PTX" as Figure 10. From Table 10, we find that including the information of "PTX" leads to an MAPE reduction from 0.190 to 0.183 and an RMSE reduction from 23.921 to 21.050, and it increases the R^2 from 0.845 to 0.853. However, because "PTX" is the projected taxi demand, the performance of the model "BPNN + T + E + TX" is better than the model "BPNN + T + E + PTX". Furthermore, Figure 11 indicates that the performance of the model "BPNN + T + E + PTX" is better than the model "BPNN + T + E" for most hours.

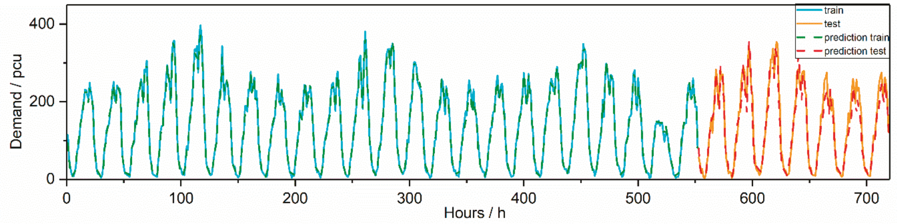
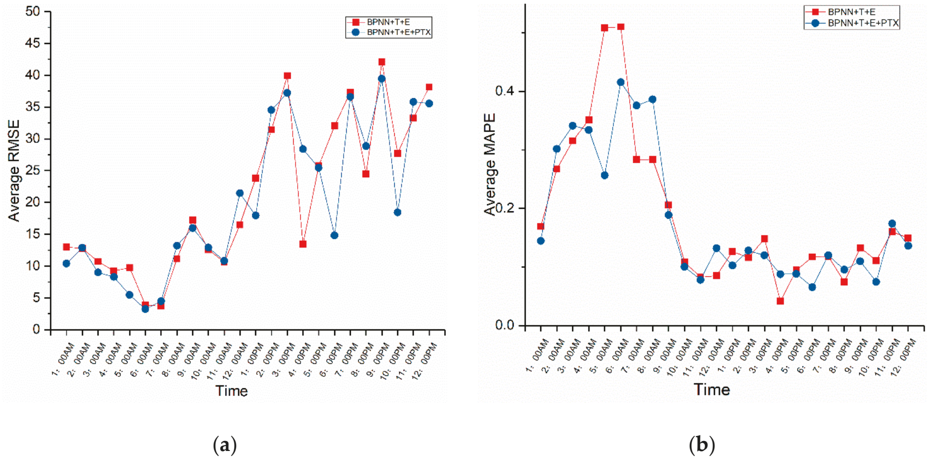


Figure 10. The demand prediction of online taxi-hailing based on the model "BPNN + T + E + PTX".

Table 10. Prediction effects of model "BPNN + T + E", "BPNN + T + E + PTX" and "BPNN + T + E + TX".

Models	Training Set			Testing Set		
	MAPE	RMSE	R^2	MAPE	RMSE	R^2
BPNN + T + E	0.178	23.692	0.851	0.190	23.921	0.845
BPNN + T + E + PTX	0.169	19.325	0.869	0.183	21.050	0.853
BPNN + T + E + TX	0.127	18.632	0.893	0.132	18.696	0.866



(**a**) (**b**)

Figure 11. Prediction effects of online taxi-hailing demand predictions in different hours. (**a**) Average RMSE of online taxi-hailing demand predictions in different hours; (**b**) Average MAPE of online taxi-hailing demand predictions in different hours.

6. Discussion and Conclusions

6.1. Discussion

We proposed a real-time prediction method of online taxi-hailing demand and studied the impacts of forecasting taxi demand on the accuracy of online taxi-hailing demand. Then, we obtained the findings below:

1. The demand for taxi and online taxi-hailing is regular in the Bell Tower area, and taxi demand decreases while the online taxi-hailing demand increases in peak hours. This is because online taxi-hailing is more convenient to obtain than taxis in peak hours;
2. Based on the BPNN methods, including information "E" leads to an MAPE reduction from 0.224 to 0.190 while it decreases RMSE from 28.576 to 23.921, and it increases the R^2 from 0.819 to 0.845. Likewise, including information "TX" leads to an MAPE reduction from 0.190 to 0.132, and it increases the R^2 from 0.845 to 0.866. Figure 7a shows that model "BPNN + T + E + TX" obtains the lowest MAPE among three predictions except at 6 a.m., 8 a.m. and 9 p.m., and from Figure 8, we know that model "BPNN + T + E + TX" obtains the lowest RMSE among three predictions except at 4, 7, and 9 p.m.;
3. Based on the XGB methods, including information "E" leads to an MAPE reduction from 0.333 to 0.197 while it reduces RMSE from 26.296 to 21.206, and it increases the R^2 from 0.833 to 0.857. And including information "TX" leads to an MAPE reduction from 0.197 to 0.139, and it increases the R^2 from 0.857 to 0.865. Figure 7b indicates that the performance of the model "XGB + T + E + TX" is the best, except at 11 a.m. and 5 p.m., and from Figure 8 we know that performance of model "XGB + T + E + TX" is the best except at 11 a.m., 12 a.m., 4 p.m. and 5 p.m.;
4. We found that the model "BPNN + T + E + TX" is available for forecasting online taxi-hailing demand. The performance of the model, which includes information of "PTX", is better than the model "BPNN + T + E", but its accuracy is lower than the model "BPNN + T + E + TX". Compared with the model "BPNN + T + E", including the information of "PTX" leads to a reduction of MAPE from 0.190 to 0.183 and an RMSE reduction from 23.921 to 21.050, and it increases R^2 from 0.845 to 0.853. This indicates that considering the information of "PTX" improves the predictability of online taxi-hailing demand. However, due to a reduction in the accuracy of "PTX", the performance of the model "BPNN + T + E + TX" is better than the model "BPNN + T + E + PTX".

However, the research still has some limitations. In the future, these limitations should be studied. For example, we did not use linear regression models to predict online taxi-hailing demand. Moreover, we should propose a method to forecast multiple trip demands simultaneously. Additionally, we will set projected environmental data as factors of real-time demand prediction for real-time forecasting.

6.2. Conclusions

In this research, the data-driven forecasting method of online taxi-hailing demand is carried out. To improve the prediction effects of online taxi-hailing demand, we proposed two methods for predicting online taxi-hailing demand based on BPNN and XGB, respectively. Then, we tested the two methods considering the information of "T", "E" and "TX". The results indicate that considering more information could improve the prediction accuracy of the models. Next, we forecasted the taxi demand and introduced a real-time online taxi-hailing demand forecasting method based on the projected taxi demand. We found that including the information of "PTX" improved prediction performance of model "BPNN + T + E". Furthermore, MAPE, RMSE and R^2 of the testing set of the model "BPNN + T + E + PTX" are, respectively, improved to 0.183, 21.050, and 0.853. Because the more precise traffic demand forecasting method can provide a more reasonable basis for public resources' dispatch, the proposed method is cost-effective in the intelligent transportation system.

Furthermore, more experiments about traffic demand prediction can be considered. For instance, the multi-mode traffic demand predictor could be proposed to improve the prediction accuracy by considering the interaction among multiple modes of transportation. Meanwhile, the multi-mode traffic demand predictor can also take the interaction among different regions into account.

Author Contributions: Conceptualization, Z.L. and H.C. (Hong Chen); methodology, Z.L.; software, Z.L.; validation, X.S., H.C. (Hengrui Chen) and Z.L.; formal analysis, Z.L.; investigation, Z.L.; resources, Z.L.; data curation, H.C. (Hong Chen); writing—original draft preparation, Z.L.; writing—review and editing, Z.L.; visualization, H.C. (Hong Chen); supervision, H.C. (Hong Chen); project administration, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Technology Project of the Shaanxi Transportation Department, grant number 15-39R.

Acknowledgments: This study was supported by the Technology Project of Shaanxi Transportation Department.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chang, H.-W.; Tai, Y.-C.; Hsu, J.Y.-J. Context-aware taxi demand hotspots prediction. *Int. J. Bus. Intell. Data Min.* **2010**, *5*, 3–18. [CrossRef]
2. Moreira-Matias, L.; Gama, J.; Ferreira, M.; Damas, L. A predictive model for the passenger demand on a taxi network. In Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, AK, USA, 16–19 September 2012; pp. 1014–1019.
3. Moreira-Matias, L.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; Damas, L. On Predicting the Taxi-Passenger Demand: A Real-Time Approach. In *Portuguese Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 54–65.
4. Moreira-Matias, L.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; Damas, L. Predicting Taxi–Passenger Demand Using Streaming Data. *IEEE Trans. Intell. Trans. Syst.* **2013**, *14*, 1393–1402. [CrossRef]
5. Zhang, K.; Feng, Z.; Chen, S.; Huang, K.; Wang, G. A Framework for Passengers Demand Prediction and Recommendation. In Proceedings of the 2016 IEEE International Conference on Services Computing (SCC), San Francisco, CA, USA, 27 June–2 July 2016; pp. 340–347.
6. Jagannathan, N.D.G.R.K. A Multi-Level Clustering Approach for Forecasting Taxi Trip demand. In Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 223–228.
7. Peng, X.; Pan, Y.; Luo, J. Predicting high taxi demand regions using social media check-ins. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 2066–2075.
8. Zhao, K.; Khryashchev, D.; Freire, J.; Silva, C.; Vo, H. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 833–842.
9. Xu, J.; Rahmatizadeh, R.; Boloni, L.; Turgut, D. A Sequence Learning Model with Recurrent Neural Networks for Taxi Demand Prediction. In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Singapore, 9-12 October 2017; pp. 261–268.
10. Zhang, D.; He, T.; Lin, S.; Munir, S.; Stankovic, J.A. Taxi-Passenger-Demand Modeling Based on Big Data from a Roving Sensor Network. *IEEE Trans. Big Data* **2017**, *3*, 362–374. [CrossRef]
11. Bao, Y.; Sun, Y.-E.; Bu, X.; Du, Y.; Wu, X.; Huang, H.; Luo, Y.; Huang, L. How Do Metro Station Crowd Flows Influence the Taxi Demand Based on Deep Spatial-Temporal Network? In Proceedings of the 2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), Shenyang, China, 6–8 December 2018; pp. 188–192.
12. Davis, N.; Raina, G.; Jagannathan, K. Taxi Demand Forecasting: A HEDGE-Based Tessellation Strategy for Improved Accuracy. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 3686–3697. [CrossRef]
13. Markou, I.; Rodrigues, F.; Pereira, F.C. Real-Time Taxi Demand Prediction using data from the web. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 1664–1671.

14. Ishiguro, S.; Kawasaki, S.; Fukazawa, Y. Taxi Demand Forecast Using Real-Time Population Generated from Cellular Networks. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers—UbiComp '18, Singapore, 8–12 October 2018; pp. 1024–1032.
15. Liao, S.; Zhou, L.; Di, X.; Yuan, B.; Xiong, J. Large-scale short-term urban taxi demand forecasting using deep learning. In Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), Jeju, Korea, 22–25 January 2018; pp. 428–433.
16. Vanichrujee, U.; Horanont, T.; Pattara-Atikom, W.; Theeramunkong, T.; Shinozaki, T. Taxi Demand Prediction using Ensemble Model Based on RNNs and XGBOOST. In Proceedings of the 2018 International Conference on Embedded Systems and Intelligent Technology & International Conference on Information and Communication Technology for Embedded Systems (ICESIT-ICICTES), Khon Kaen, Thailand, 7–9 May 2018; pp. 1–6.
17. Xu, J.; Rahmatizadeh, R.; Boloni, L.; Turgut, D. Real-Time Prediction of Taxi Demand Using Recurrent Neural Networks. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 2572–2581. [CrossRef]
18. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Chuxing, D.; Li, Z. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 2588–2595.
19. Zhang, W.; Ukkusuri, S.; Yang, C. Modeling the Taxi Drivers' Customer-Searching Behaviors outside Downtown Areas. *Sustainability* **2018**, *10*, 3003. [CrossRef]
20. Yan, H.; Zhang, Z.; Zou, J. An online spatio-temporal model for inference and predictions of taxi demand. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 3550–3557.
21. Kuang, L.; Yan, X.; Tan, X.; Li, S.; Yang, X. Predicting Taxi Demand Based on 3D Convolutional Neural Network and Multi-task Learning. *Remote Sens.* **2019**, *11*, 1265. [CrossRef]
22. Liu, L.; Qiu, Z.; Li, G.; Wang, Q.; Ouyang, W.; Lin, L. Contextualized Spatial–Temporal Network for Taxi Origin-Destination Demand Prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3875–3887. [CrossRef]
23. Markou, I.; Kaiser, K.; Pereira, F.C. Predicting taxi demand hotspots using automated Internet Search Queries. *Transp. Res. Part C Emerg. Technol.* **2019**, *102*, 73–86. [CrossRef]
24. Qiu, Z.; Liu, L.; Li, G.; Wang, Q.; Xiao, N.; Lin, L. Taxi Origin-Destination Demand Prediction with Contextualized Spatial-Temporal Network. In Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 8–12 July 2019; pp. 760–765.
25. Rodrigues, F.; Markou, I.; Pereira, F.C. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Inf. Fusion* **2019**, *49*, 120–129. [CrossRef]
26. Terroso-Saenz, F.; Munoz, A.; Cecilia, J.M. QUADRIVEN: A Framework for Qualitative Taxi Demand Prediction Based on Time-Variant Online Social Network Data Analysis. *Sensors* **2019**, *19*, 4882. [CrossRef] [PubMed]
27. Xu, Y.; Li, D. Incorporating Graph Attention and Recurrent Architectures for City-Wide Taxi Demand Prediction. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 414. [CrossRef]
28. Yu, H.; Chen, X.; Li, Z.; Zhang, G.; Liu, P.; Yang, J.; Yang, Y. Taxi-Based Mobility Demand Formulation and Prediction Using Conditional Generative Adversarial Network-Driven Learning Approaches. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3888–3899. [CrossRef]
29. Liu, X.; Sun, L.; Sun, Q.; Gao, G. Spatial Variation of Taxi Demand Using GPS Trajectories and POI Data. *J. Adv. Trans.* **2020**, *2020*, 7621576. [CrossRef]
30. Saadallah, A.; Moreira-Matias, L.; Sousa, R.; Khiari, J.; Jenelius, E.; Gama, J. BRIGHT—Drift-Aware Demand Predictions for Taxi Networks. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 234–245. [CrossRef]
31. Safikhani, A.; Kamga, C.; Mudigonda, S.; Faghih, S.S.; Moghimi, B. Spatio-temporal modeling of yellow taxi demands in New York City using generalized STAR models. *Int. J. Forecast.* **2020**, *36*, 1138–1148. [CrossRef]
32. Liu, Z.; Chen, H.; Li, Y.; Zhang, Q. Taxi Demand Prediction Based on a Combination Forecasting Model in Hotspots. *J. Adv. Trans.* **2020**, *2020*, 1302586. [CrossRef]
33. Wang, Z.; Luo, P.; Cheng, L.; Zhang, S.; Shen, J. Hapten-antibody recognition studies in competitive immunoassay of alpha-zearalanol analogs by computational chemistry and Pearson Correlation analysis. *J. Mol. Recognit.* **2011**, *24*, 815–823. [CrossRef]

34. Rajabi-Kiasari, S.; Hasanlou, M. An efficient model for the prediction of SMAP sea surface salinity using machine learning approaches in the Persian Gulf. *Int. J. Remote Sens.* **2020**, *41*, 3221–3242. [CrossRef]
35. Wang, J.-Z.; Wang, J.-J.; Zhang, Z.-G.; Guo, S.-P. Forecasting stock indices with back propagation neural network. *Expert Syst. Appl.* **2011**, *38*, 14346–14355. [CrossRef]
36. Selbesoglu, M.O. Spatial Interpolation of GNSS Troposphere Wet Delay by a Newly Designed Artificial Neural Network Model. *Appl. Sci.* **2019**, *9*, 4688. [CrossRef]
37. Karsoliya, S. Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture. *Int. J. Eng. Trends Technol.* **2012**, *3*, 714–717.
38. Liu, Y.; Jiang, W.; Zhang, X. Research on Optimized Energy Scheduling of Rural Microgrid. *Appl. Sci.* **2019**, *9*, 4641. [CrossRef]

applied sciences

MDPI

Article

Heatwave Damage Prediction Using Random Forest Model in Korea

Minsoo Park [1], Daekyo Jung [2], Seungsoo Lee [2] and Seunghee Park [1,3,*]

[1] School of Civil, Architectural Engineering & Landscape Architecture, Sungkyunkwan University, Suwon 16419, Korea; pms5343@skku.edu

[2] Department of Convergence Engineering for Future City, Sungkyunkwan University, Suwon 16419, Korea; jdaekyo@skku.edu (D.J.); skklss@skku.edu (S.L.)

[3] Technical Research Center, Smart Inside Co., Ltd., Suwon 16419, Korea

* Correspondence: shparkpc@skku.edu

Received: 31 October 2020; Accepted: 19 November 2020; Published: 20 November 2020

Abstract: Climate change increases the frequency and intensity of heatwaves, causing significant human and material losses every year. Big data, whose volumes are rapidly increasing, are expected to be used for preemptive responses. However, human cognitive abilities are limited, which can lead to ineffective decision making during disaster responses when artificial intelligence-based analysis models are not employed. Existing prediction models have limitations with regard to their validation, and most models focus only on heat-associated deaths. In this study, a random forest model was developed for the weekly prediction of heat-related damages on the basis of four years (2015–2018) of statistical, meteorological, and floating population data from South Korea. The model was evaluated through comparisons with other traditional regression models in terms of mean absolute error, root mean squared error, root mean squared logarithmic error, and coefficient of determination (R^2). In a comparative analysis with observed values, the proposed model showed an R^2 value of 0.804. The results show that the proposed model outperforms existing models. They also show that the floating population variable collected from mobile global positioning systems contributes more to predictions than the aggregate population variable.

Keywords: heatwaves; big data; random forest regression model; machine learning; prediction

1. Introduction

According to the National Center for Environmental Information of the National Oceanic and Atmospheric Administration, the average annual global temperature has reached an all-time high over the past five years (0.75–0.95 °C rise from the average annual temperature in the 20th century) and is continuing to gradually increase. Global warming has considerably changed the climate in recent decades, increasing the probability and intensity of meteorological and climatic disasters [1,2]. The duration and intensity of heatwaves are expected to increase with an increase in the average annual temperature, and deaths from heatwaves are expected to double [3]. The record heatwave in the United Kingdom in 2003, which killed 70,000 people, is expected to become normal summer weather by 2040 [4].

Because heatwaves cause human and physical disasters every year, it is important to minimize disaster damage by establishing timely and preemptive disaster responses. A disaster response is a continuous decision making process conducted on the basis of a variety of information and past experiences that are continuously gathered from a range of locations. Further, disaster response is conducted from the moment a disaster is perceived to have occurred until the time when it ends. In the past, data collection techniques were less effective and provided limited information for use in contextual judgment and decision making. Consequently, owing to the lack of information available for contextual judgment and decision making, disaster responses were highly dependent on the subjective experiences of decision makers. Furthermore, although data collection technology has developed rapidly with an increase in the information available for decision making, the capability of humans to process and use this information in disaster response is limited, especially in cases that require swift decision making.

The importance of utilizing big data and artificial intelligence (AI)-based analysis for the rapid processing of various types of data has been recognized. Big data refers to large and diverse forms of data that cannot be processed by traditional database systems. Further, big datasets can include signals, images, and documents whose sizes increase exponentially; such data are abundant, owing to the development of sensing and social media-oriented communication technologies within the present Internet of Things environment [5,6]. Big data systems not only utilize a variety of data quickly but are also expected to play a crucial role in analyzing meaningful information. However, early systems only focused on data collection and storage [7]. To produce meaningful results from big data, AI technology as well as simple statistical and visualization functions must be employed for analysis and prediction.

Heatwave definitions vary among different countries [8]; however, heatwaves are generally defined on the basis of the normal weather and temperatures corresponding to the seasons of a region, and they are said to occur when there is a large deviation from the normal climate pattern in a given region. These extreme weather conditions occur locally and extensively, which limits rapid disaster response. In particular, because such extreme weather conditions occur extensively throughout a region, response procedures, such as preparing resources immediately in the event of a disaster, are limited. This indicates the need to develop early warning systems to guide disaster responses. Previous studies have focused on mortality as an endpoint for the analysis of damage caused by heatwaves [9–11], and only few studies have focused on morbidity as an indicator [12]. In addition, most studies have adopted only weather-related parameters as predictor variables of mortality. However, even under the same weather conditions, the damage pattern can vary, and it depends on other variables, such as the vulnerable population. This emphasizes the need to consider various variables as well as weather-related parameters to predict heatwave damage.

In this study, a heat-related health prediction model was developed on the basis of a machine learning algorithm for early warning systems. The purpose of this study is to help decision makers to preemptively respond, reducing human and economic losses. This paper is organized as follows: Section 2 describes the architecture of the random forest (RF) architecture and variables that can represent damage caused by heatwaves obtained from a big data collection site operated in South Korea. Experimental results, including variable evaluation, model optimization, and RF's accuracy evaluation in comparison with the tradition regression models is mention on Section 3. A trained model was applied to the site and visualized—this is also specified in Section 3. Section 4 presents the discussion and conclusion of this study.

2. Methodology

2.1. Test Area

South Korea was selected as the test bed, and its heatwave characteristics were investigated to establish the range and duration of the collected data for model training. The typical weather pattern that causes heatwaves in South Korea is a significant rise in temperature during the daytime, owing to stagnant high atmospheric pressure, which is a widespread occurrence across the country [13]. Although heatwave standards vary by country, a heatwave warning in South Korea is issued when the daily maximum temperature is expected to be above 33 °C for at least 2 consecutive days. Alerts are concentrated mainly from June to August. Heatwave occurrences in South Korea exhibit substantial interannual variability, but recently, they have become more frequent in late May and early September, and their frequency and intensity have increased [14,15]. In particular, record-breaking heatwaves occurred in 2016 and 2018, causing many casualties [16]. The Korea Disease Control and Prevention

Agency (KDCA; formerly the Korea Centers for Disease Control and Prevention) has been operating a nationwide thermal disease monitoring system since 2011 to determine the weekly health damage caused by heatwaves from late May to early September every year. South Korea has 17 administrative districts composed of 8 municipalities and 9 provinces. In accordance with the characteristics of these test beds, we set the range resolution to match the 17 administrative divisions, and the temporal resolution was set to a 1-week period to match the disease monitoring system data from the KDCA.

2.2. Variable Selection

Relevant variables were selected to predict heat-related damage. Heat-related diseases mainly occur in the form of cardiovascular and respiratory diseases and heatstroke [17]; consequently, various epidemiological studies of their occurrence have been conducted worldwide [17–19]. Among the most important characteristics of the damage caused by heatwaves and the corresponding vulnerabilities are the damage patterns of disasters, which cannot be obtained from temperature variables alone [20,21]. Studies have shown that the damage caused by disasters is related to geographic features [22,23], surface relative humidity [24–26], wind speed [27,28], population density [29], economic status [30], and vulnerable occupational groups (laborers, construction, and agricultural workers) [31–34]. On the basis of important characteristics determined in previous studies, we selected the following variables: temperature, humidity, wind speed, number of vulnerable occupational groups, insurance premiums per person, personal income per person, floating population, and registered population of residents (the number of people counted by the administration). The vulnerable population can be inferred from data on insurance premiums, income, and vulnerable occupational groups; further, as the values of these indicators increased, the number of patients with thermal diseases increased. However, both the aggregate and floating populations were used as population variables, and it was expected that the floating population, which reflects real-time information, would be a more useful variable for predictions than the aggregate population.

2.3. Random Forest Regression

RF is an ensemble machine learning method that combines several separately trained models to create a strong learner that can be applied for classification and regression [35]. Such a combination of individual models can reduce overfitting and improve generalization. Therefore, RF has the advantages of high prediction accuracy and algorithm robustness. When training ensemble classifiers, techniques involving the use of different datasets or properties are applied to create different training models. As shown in Figure 1, RF is based on the bootstrap method which is resampling technique that involves random sampling of a dataset with replacement. Then repeats the process k times to obtain several independent and identically distributed training subsets $\{S_{train,1}, S_{train,2}, \ldots, S_{train,\mathrm{k}}\}$, which have n samples. Then, m features from the n samples are selected without accepting duplicate samples. Prediction results from different decision trees build each training subset. The most commonly obtained forecast results are selected and determined by the final forecast [35,36]. In conclusion, although some trees created in RF may be exposed to overfitting, overfitting of the RF can be prevented by generating a large number of trees. RF algorithms have been applied to various disaster fields to predict [37,38], forecast, and evaluate risks [39,40].

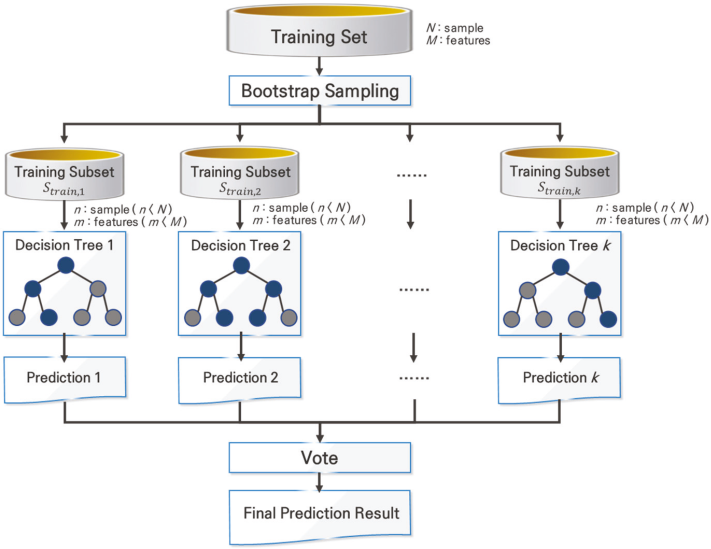


Figure 1. Architecture of a random forest.

A loss function measures the similarity between the values predicted by a model and the correct values. To increase the accuracy of a model, the loss should be reduced as the model is trained. Different loss functions are used depending on the characteristics of the model (classification or regression) and dataset. The representative loss functions for measuring errors in regression models are mean absolute error (MAE) and mean squared error (MSE):

$$MAE = \frac{\sum_{i=1}^{N} \left| y_i - \hat{y} \right|}{N} \tag{1}$$

$$MSE = \frac{\sum_{i=1}^{N} (y_i - \hat{y})^2}{N} \tag{2}$$

where N is the total number of data points, y is the real (observed) output value, and $\hat{y}$ is the predicted output value. When determining the MAE, the difference between the observed and predicted values of each data point is summed, and when determining the MSE, the square of the difference between observed and predicted values is summed. Therefore, the MSE is more sensitive to outlier values than the MAE. When the temperature exceeds a certain range, the heatstroke patients with thermal damage is characterized by a rapid increase in the incidence of patients. Consequently, MAE was considered as a loss function in this study to apply the characteristic of the target data.

To evaluate regression models, the proximity of predicted values to the observed data is quantified on the basis of the MAE, root mean squared error (RMSE), root mean squared logarithmic error (RMSLE), and coefficient of determination (R^2), which are mainly used to evaluate accuracy [41,42]. However, the mean deviations of MAE, RMSE, and RMSLE (the lower the value, the higher the accuracy) have different values depending on the scale; therefore, it is difficult to make inferences using the absolute values alone. In contrast, R^2 is a relative value because it is the variance ratio of dependent variables predicted from independent variables; thus, the performance can be intuitively determined. R^2 generally ranges from 0 to 1. Note that if the R^2 value of a model is 0.7 or more, the model is usually considered reasonable [43].

The RF model was established to predict the number of patients with heat-related diseases caused by heatwaves. Socioeconomic, demographic, meteorological, and demographic data were collected and used as input variables for the model. The Boruta algorithm was used to filter the variables in the RF model [38]; this algorithm uses a Z score calculated by dividing the average loss by its standard deviation. It was implemented using an R package [44] to confirm whether certain variables can be used as predictive model inputs. Typical parameters of the random forest algorithm are ntree and max depth. To select optimal hyperparameters, the minimum loss function value (MAE) was found by increasing the number of decision trees (n-tree) and their maximum depth (max depth). After separating the dataset comprising the selected variables into training and test datasets, we evaluated the model trained using the training dataset by comparing it with other traditional regression models. Finally, the mean decrease in impurity (i.e., Gini importance) was used to extract the variable importance values, i.e., to determine the predicted contribution of each variable's model.

3. Results of Predicting the Number of Heatwave-Related Patients

3.1. Data Collection and Pre-Processing for Model Training

The variables and target data are listed in Table 1 with their data sources and renewal cycles. The variables are categorized as static or dynamic. Further, the abbreviations of the variables are used hereafter in the main text, figures, and tables. The static variables were pre-collected from a government agency that manages big data. They are universally updated quarterly and yearly, making them less volatile when predicting the number of heatwave-related patients in summer. In contrast, the dynamic variables, such as floating population and weather information, change with time. In South Korea, big data regarding the floating population are estimated on the basis of mobile big data collected hourly and monthly by SK Telecom's nationwide mobile communication base stations, and the estimated data are obtained from the Statistical Data Center. They are also estimated using public big data and communication data provided by the Seoul Open Data Plaza. Weather data are collected hourly and were provided by the Korea Meteorological Administration (KMA).

Table 1. Descriptions of variables to predict the number of heatwave-related patients.

Variable Description	Abbreviation	Units	Data Source
Static variables—socioeconomic and demographic data			Korean statistical information service
Per capita income	Income	×$1000	
Insurance premiums per person	Insurance	×$1000	
Resident registration population	RRP	×1	
Number of vulnerable occupational groups (agricultural, manufacturing, and construction workers)	V-groups	×1000	
Dynamic variables—meteorological data			KMA
Maximum temperature of the week	Max Tem	°C	
Minimum temperature of the week	Min Tem	°C	
Mean temperature of the week	Mean Tem	°C	
Median temperature of the week	Median Tem	°C	
Variance temperature of the week	Variance Tem	°C	
Mean humidity of the week	Mean Hum	%	
Mean wind speed of the week	Mean wind speed	m/s	
Dynamic variables—demographic data			Statistical data center
Floating population	FP	×1	

However, the weather data, particularly those collected from sensors, may have missing values due to sensor defects. To address the problem of missing values, we used datasets consisting of columns with no missing values in order to predict the missing values of other datasets. Target data were based on weekly data obtained from the thermal disease monitoring system managed by KDCA (patients with heat-related diseases and deaths caused by heatwaves in emergency rooms nationwide);

data regarding heat-related diseases such as heat stroke, exhaustion, cramps, fainting, and edema were also provided as weekly data. The resolution of the entire dataset was unified through considering the data properties of the features and targets; the temporal resolution was set to 1 week, and the range resolution was set on the basis of the South Korean administrative divisions. The datasets were randomly used for classification—80% were used as learning data and the remaining 20% as test data. Finally, variables were normalized before being inputted into the RF model to avoid creating a model that depends on specific variable units owing to the different ranges of each variable.

All variables were confirmed using the Boruta algorithm. The contribution of each variable to the RF prediction model is shown in Figure 2. The edges of each box represent the quartiles, and the line through each box represents the median. Each bar represents the 1.5 interquartile range of the nearer quartile, and the open circles represent outliers. The blue boxes correspond to the minimal, average, and maximum Z scores of a shadow attribute in the Boruta algorithm. The green boxplots correspond to confirmed important attributes. It was confirmed that all collected data from the Boruta algorithm can be used as variables of the predictive model.

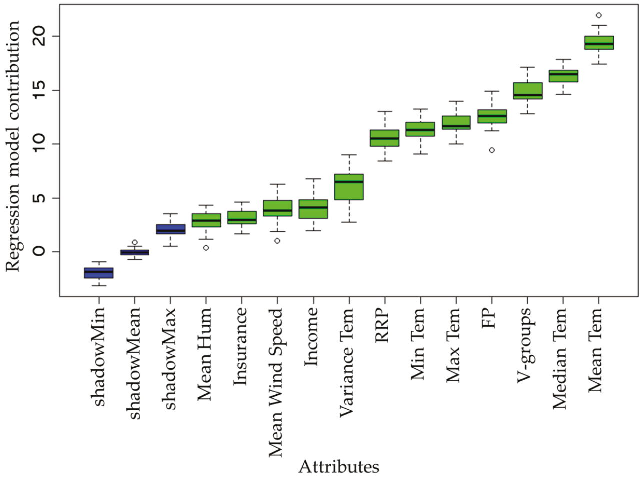
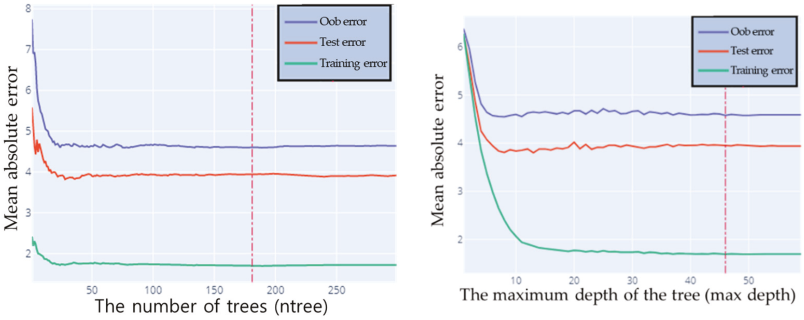


Figure 2. Contribution ranking importance of the 12 independent variables in the random forest (RF)-based variable reduction algorithm from the Boruta package [44] in R.

3.2. Hyper-Parameter Optimization

The experiment was conducted using the Scikit-learn (v.0.22.2) Python package [45] to implement the RF; the hardware platform was an Intel (R) Core (TM) i9-9900k 3.60 GHz CPU with 32 GB of RAM. The out-of-bag (OOB) error is mainly used to measure errors in machine learning models, such as bootstrap aggregation (bagging), which can be substituted for test errors [46]. The lowest MAE was found for the training, OOB, and test errors as n-tree and max depth increased, and is shown in Figure 3. When the number of decision trees was more than 100, all graphs remained almost unchanged, and when the number of decision trees was 181, the lowest OOB error was found (4.59). The training and test errors at this time were 1.67 and 3.94, respectively.



(**a**) (**b**)

Figure 3. Hyperparameter optimization in the RF regression model: (**a**) training curves with respect to number of trees and (**b**) training curves with respect to maximum depth of trees.

On the other hand, when the number of decision trees was 181, the maximum depth of the decision tree remained constant from over 40, and the lowest OOB error (4.58) was found when the depth was 46. The train error and test error at this time were 1.69 and 3.94, respectively. Therefore, the hyperparameters were determined with 181 decision trees and 46 tree depths.

3.3. Model Comparasion

The RF model was trained on the basis of the determined hyperparameters, and test data were applied to the regression model. The linear regression relationship between the predicted data from the model and test data is shown in Figure 4. The black line in the graph represents the regression line. The x-axis represents the weekly predicted number of patients with heat-related diseases in a specific region, as predicted by the model, and the y-axis indicates the weekly number of real patients with heat-related diseases in the region. The translucent band around the regression line area indicates the size of the confidence interval, which was 95% in this case. The red dotted line indicates when the model accurately reflected reality (slope: 1). The linear fitting slope of this RF model was 1.11. In particular, when high values were predicted, they tended to be underestimated compared to the observed values; however, the models were confirmed to be relatively reasonable.

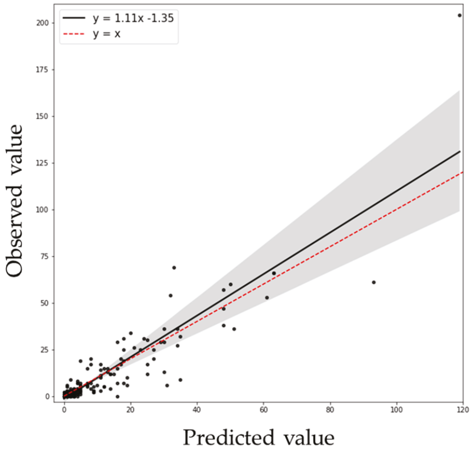


Figure 4. Linear fitting results of test data.

To compare the accuracy of the regression model more quantitatively, Table 2 compares its results with those of other regression models. In particular, the RF model is compared with linear regression, decision tree, and support vector machine (SVM) models. All models were trained using the same training set, and all the trained models were evaluated by the same test set. However, some of the values predicted by the SVM model were negative; because the values must be greater than or equal to zero, we treated all negative values as zeros. As shown in Table 2, the best values for all the considered metrics, including MAE (3.816), RMSE (8.655), RMSLE (0.645), and R^2 (0.803), were obtained for the RF model. This means that the RF is more accurate than other models for making predictions, and the R^2 value of 0.803 proves that this model is reasonable.

Table 2. Comparisons of performance evaluation.

Method	MAE	RMSE	RMSLE	R^2
Logistic regression	5.301	12.460	0.855	0.593
SVM	5.184	8.800	0.956	0.797
Decision tree	5.524	13.384	0.803	0.531
Random forest	**3.816**	**8.655**	**0.645**	**0.804**

The bold is the best result among other methods.

3.4. Feature Importance

Figure 5 shows the estimated variable importance rankings corresponding to the model. The weekly mean temperature variable, which had a value of 0.440, contributed the most in this model, followed by the vulnerable occupational groups (0.129), weekly median temperature (0.102), floating population (0.098), and weekly max temperature (0.085) variables. These five variables can be considered the main variables for prediction, whereas the rest are less important. Interestingly, the variable importance rankings proved that the floating population variable, which changes with time, had a greater effect on prediction than the population of registered residents. However, regional economic indicators had less impact on diseases related to heatwaves, as observed from the low values for income (0.020) and insurance (0.013).

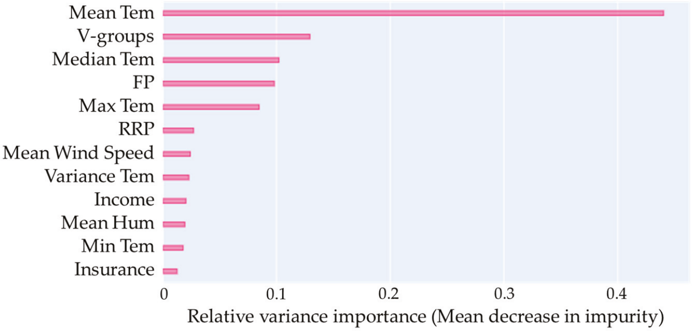


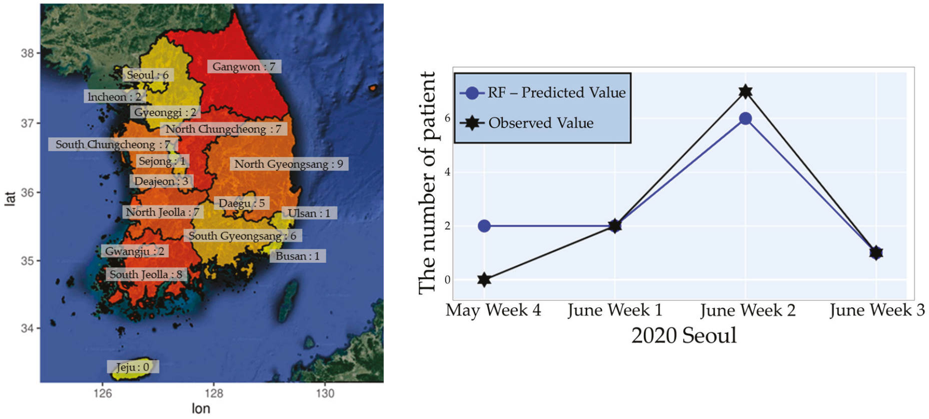
Figure 5. Variance importance in the RF model.

3.5. Model Application and Visualization

To apply the validated model, we predetermined the dynamic variables from predictions. Because the time series of variables and result values were the same, the predicted variable values must be used for prediction. With regard to static variables, we employed the latest data as inputs among the information that is updated periodically, which is the same as in model learning. The values of the dynamic population were replaced with dynamic variables using weather forecast data provided by

KMA on a weekly basis and a time series forecasting library, called Prophet [47], which is provided by Facebook.

The performance results and visualization of the model are shown in Figure 6. From the end of May, which was when heatwave management began, the substituted variables were inputted into the model for 4 weeks, and then the predicted values were obtained and compared with the observed values obtained from KDCA on the weekend. The forecasted and observed values for Seoul were compared for 4 weeks, and in the second week of June, the predicted values for each administrative district of South Korea were numerically quantified to visualize the high-risk areas and provide information to heatwave disaster response decision makers. The high-risk areas are shown in dark colors, whereas the lower risk areas are shown in lighter colors. Considering objectivity by region, we used the number of patients and the predicted floating population ratios to calculate the risk. Four weeks of data were applied to real-world situations, resulting in an R^2 value of 0.70.



(**a**) (**b**)

Figure 6. (**a**) Visualization of predicted number of heatwave-related patients in the second week of June 2020 in South Korea. (**b**) Predicted and observed data for number of heatwave-related patients in Seoul over a month.

4. Discussion and Conclusions

Heatwave damage prediction has been investigated in the United States, Europe, and Asia [9,48–50]. However, existing predictions are limited to practical notification systems owing to unrepresentative data and insufficient data accuracy [51]. According to previous research, this problem is due to the use of heatwave mortality alone as the endpoint of damage. Because the mortality rate of heatwaves is exceedingly small compared to that of the general population, it is more effective to predict risk by morbidity, which is relatively higher in proportion than heatwave mortality.

"Temperatures exceeding 33 °C" is the only available criterion for identifying the danger of heatwaves in South Korea, which allows the government to raise risk awareness by alerting the public. However, the damage to the population (deaths and sickness) caused by heatwaves varies even at the same temperature. Therefore, using only temperature data cannot determine the level of damage to peoples' health. On the basis of epidemiological investigations performed in previous research, we selected relevant variables and evaluated them by the Boruta algorithm. Then, a random forest-based heatwave damage prediction method was proposed, and its performance was compared with other traditional models. Previous studies considering demographic information have mainly used data with static characteristics, such as monthly statistical information. More accurate predictions were achieved by matching the exposure to heatwaves in a specific area to the population in that area using dynamic population data updated more frequently, allowing this variable to contribute more to the prediction.

In the evaluation of the importance of variables, the average temperature variable and the number of occupational groups that are considered to be vulnerable to heat waves were highly evaluated (the average temperature for a week is the sum of the week, that is, the accumulated temperature). This result supports the importance of predicting the cumulative temperature in advance and responding in advance in order to minimize heat damage. In addition, it provides grounds that preemptive responses from the government, such as operation of sprinkler trucks and installation of shade curtains, should be made in areas with many vulnerable occupations. The learning process performed to build the machine learning model used independent variables based on previously recorded data. However, it is difficult to apply big data to a real-time environment owing to limitations such as irregularity in the frequency of data. Furthermore, when the time series of the dependent and independent variables are configured identically in training, the conditions for predictions are not effectively established in practical systems. Therefore, a method that employs predicted variable values was proposed. Although the accuracy of predicting future patients by applying predicted data is lower than the test accuracy during validation, the R^2 value of 0.70 supports the fact that this model provides reasonable information. Governments can use the methods developed in this study to provide disaster response decision makers with a reasonable basis for prioritizing an administrative area to provide a preemptive response and disaster support.

Nonetheless, this study has several limitations. First, the temporal resolution of the predicted values is relatively coarse-grained; thus, it is impossible to provide daily predictions for heatwaves, which are expected to occur every day. Most studies on heatwaves in South Korea have provided weekly information [11,49]. As a result, it was inferred that the resolution of target values (heat patients) is tailored to the minimum information time unit of data source. Secondly, during the experiment, data obtained across the country were input into one learning algorithm, and regional differences between administrative districts were not considered. South Korea is a relatively small country; although the regional environmental difference is relatively smaller than that in larger countries, the differences in geography and weather between its eastern and western regions are substantial. Therefore, developing individual algorithms for each region can improve model performance. This problem can be solved because the model can be trained for each region if sufficient datasets are available. Since 2018, South Korea has been managing vulnerable populations by operating heatwave shelters. The prediction model established in this study will contribute to future studies to select regions at risk of heatwaves and provide decision makers with a basis for installing heat shelters using high regional resolutions and estimating the cumulative number of patients relative to the population.

Author Contributions: Idea development and original draft writing, M.P.; project administration, D.J.; draft review and editing, S.L.; supervision and funding acquisition, S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a grant (2019-MOIS31-011) from the Fundamental Technology Development Program for Extreme Disaster Response funded by the Ministry of Interior and Safety, Korea, and supported by the Korea Ministry of Land, Infrastructure and Transport (MOLIT) as an Innovative Talent Education Program for Smart City.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Djalante, R. Key assessments from the IPCC special report on global warming of 1.5 °C and the implications for the Sendai framework for disaster risk reduction. *Prog. Disaster Sci.* **2019**, *1*, 100001. [CrossRef]
2. Peduzzi, P. The disaster risk, global change, and sustainability nexus. *Sustainability* **2019**, *11*, 957. [CrossRef]
3. Schär, C.; Vidale, P.L.; Lüthi, D.; Frei, C.; Häberli, C.; Liniger, M.A.; Appenzeller, C. The role of increasing temperature variability in European summer heatwaves. *Nature* **2004**, *427*, 332–336. [CrossRef] [PubMed]
4. Public Health England. *Heatwave Plan for England*; Public Health England: London, UK, 2019.

5. Lim, J.; Bok, K.; Yoo, J. Design and implementation of a realtime public transport route guidance system using big data analysis. *J. Korea Cont. Assoc.* **2019**, *19*, 460–468.
6. Choi, S.H.; Park, Y.J.; Shim, J.H. Strengthening of disaster management ability through big data utilization. *J. Korean Soc. Civ. Eng.* **2015**, *63*, 21–28.
7. Lee, J.H.; Baek, S.H.; Lee, S.J.; Bae, H.Y. The method for Real-time complex event detection of unstructured big data. *Korea Spat. Inf. Soc.* **2012**, *20*, 99–109.
8. Meehl, G.A. More intense, more frequent, and longer lasting heat waves in the 21st century. *Science* **2004**, *305*, 994–997. [CrossRef]
9. Green, H.K.; Andrews, N.J.; Bickler, G.; Pebody, R.G. Rapid estimation of excess mortality: Nowcasting during the heatwave alert in England and Wales in June 2011. *J. Epidemiol. Comm. Health* **2012**, *66*, 866–868. [CrossRef] [PubMed]
10. Anderson, G.B.; Oleson, K.W.; Jones, B.; Peng, R.D. Classifying heatwaves: Developing health-based models to predict high-mortality versus moderate united states heatwaves. *Clim. Chang.* **2018**, *146*, 439–453. [CrossRef]
11. Kim, D.W.; Deo, R.C.; Park, S.J.; Lee, J.S.; Lee, W.S. Weekly heat wave death prediction model using zero-inflated regression approach. *Theor. Appl. Climatol.* **2019**, *137*, 823–838. [CrossRef]
12. Williams, S.; Nitschke, M.; Weinstein, P.; Pisaniello, D.L.; Parton, K.A.; Bi, P. The impact of summer temperatures and heatwaves on mortality and morbidity in Perth, Australia 1994–2008. *Environ. Int.* **2012**, *40*, 33–38. [CrossRef] [PubMed]
13. Lee, W.S.; Lee, M.I. Interannual variability of heat waves in Korea and their connection with large-scale atmospheric circulation patterns. *Int. J. Climatol.* **2016**, *36*, 4815–4830. [CrossRef]
14. Suh, M.S.; Oh, S.G.; Lee, Y.S.; Ahn, J.B.; Cha, D.H.; Lee, D.K.; Hong, S.Y.; Min, S.K.; Park, S.C.; Kang, H.S.; et al. Projections of high resolution climate changes for Korea using multiple-regional climate models based on four RCP scenarios. Part 1: Surface air temperature. *Asia Pac. J. Atmos. Sci.* **2016**, *52*, 151–169. [CrossRef]
15. Min, K.H.; Chung, C.H.; Bae, J.H.; Cha, D.H. Synoptic characteristics of extreme heatwaves over the Korean peninsula based on era interim reanalysis data. *Int. J. Climatol.* **2020**, *40*, 3179–3195. [CrossRef]
16. Lee, H.D.; Min, K.H.; Bae, J.H.; Cha, D.H. Characteristics and comparison of 2016 and 2018 heat wave in Korea. *Atmosphere* **2020**, *30*, 1–15.
17. Reid, C.E.; O'Neill, M.S.; Gronlund, C.J.; Brines, S.J.; Brown, D.G.; Diez-Roux, A.V.; Schwartz, J. Mapping community determinants of heat vulnerability. *Environ. Health Perspect.* **2009**, *117*, 1730–1736. [CrossRef]
18. Huisman, M.; Kunst, A.E.; Mackenbach, J.P. Socioeconomic inequalities in morbidity among the elderly: A European view. *Soc. Sci. Med.* **2003**, *57*, 861–873. [CrossRef]
19. Basu, R. High ambient temperature and mortality: A review of epidemiologic studies from 2001 to 2008. *Environ. Health* **2009**, *8*, 40. [CrossRef]
20. Vose, R.S.; Applequist, S.; Bourassa, M.A.; Pryor, S.C.; Barthelmie, R.J.; Blanton, B.; Bromirski, P.D.; Brooks, H.E.; Degaetano, A.T.; Dole, R.M.; et al. Monitoring and understanding changes in extremes: Extratropical storms, winds, and waves. *Bull. Am. Meteorol. Soc.* **2014**, *95*, 377–386. [CrossRef]
21. Zubov, D.; Barbosa, H.A.; Duane, G.S. A nonanticipative analog method for long-term forecasting of air temperature extremes. *arXiv* **2015**, arXiv:1507.03283.
22. Gershunov, A.; Johnston, Z.; Margolis, H.G.; Guirguis, K. The California heat wave 2006 with impacts on statewide medical emergency. *Geogr. Res. Forum* **2011**, *31*, 53–69.
23. Guirguis, K.; Gershunov, A.; Tardy, A.; Basu, R. The impact of recent heat waves on human health in California. *J. Appl. Meteor. Climatol.* **2014**, *53*, 3–19. [CrossRef]
24. Basu, R.; Samet, J.M. Relation between elevated ambient temperature and mortality: A review of the epidemiologic evidence. *Epidemiol. Rev.* **2002**, *24*, 190–202. [CrossRef] [PubMed]
25. Kovats, R.S.; Hajat, S. Heat stress and public health: A critical review. *Annu. Rev. Public Health* **2008**, *29*, 41–55. [CrossRef] [PubMed]
26. Chen, X.; Li, N.; Liu, J.; Zhang, Z.; Liu, Y. Global heat wave hazard considering humidity effects during the 21st century. *Int. J. Environ. Res. Public Health* **2019**, *16*, 1513. [CrossRef] [PubMed]
27. Lemonsu, A.; Viguié, V.; Daniel, M.; Masson, V. Vulnerability to heat waves: Impact of urban expansion scenarios on urban heat island and heat stress in Paris (France). *Urban Clim.* **2015**, *14*, 86–605. [CrossRef]
28. Li, D.; Sun, T.; Liu, M.; Wang, L.; Gao, Z. Changes in wind speed under enhance urban heat islands in the Beijing metropolitan area. *J. Appl. Meteorol. Climatol.* **2016**, *55*, 2369–2375. [CrossRef]

29. Vescovi, L.; Rebetez, M.; Rong, F. Assessing public health risk due to extremely high temperature events: Climate and social parameters. *Clim. Res.* **2005**, *30*, 71–78. [CrossRef]
30. Kim, Y.; Joh, S. A vulnerability study of the low-income elderly in the context of high temperature and mortality in Seoul, Korea. *Sci. Total Environ.* **2006**, *371*, 82–88. [CrossRef]
31. Hajat, S.; Kovats, R.S.; Lachowycz, K. Heat-related and cold-related deaths in England and Wales: Who is at risk? *Occup. Environ. Med.* **2007**, *64*, 93–100. [CrossRef]
32. Bonauto, D.; Anderson, R.; Rauser, E.; Burke, B. Occupational heat illness in Washington state, 1995–2005. *Am. J. Ind. Med.* **2007**, *50*, 940–950. [CrossRef] [PubMed]
33. Spector, J.T.; Bonauto, D.K.; Sheppard, L.; Busch-Isaksen, T.; Calkins, M.; Adams, D. A case-crossover study of heat exposure and injury risk in outdoor agricultural workers. *PLoS ONE* **2016**, *11*, e0164498. [CrossRef] [PubMed]
34. Heo, S.; Lee, E.; Kwon, B.Y.; Lee, S.; Jo, K.H.; Kim, J. Long-term changes in the heat–mortality relationship according to heterogeneous regional climate: A time-series study in Korea. *BMJ* **2016**, *6*, 1–10. [CrossRef]
35. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
36. Yao, Z.; Xu, X.; Yu, H. Floor heating customer prediction model based on random forest. In Proceedings of the 17th International Conference on Computer and Information Science, Singapore, 6–8 June 2018; pp. 573–578.
37. Dang, V.H.; Dieu, T.B.; Tran, X.L.; Hoang, N.D. Enhancing the accuracy of rainfall-induced landslide prediction along mountain roads with a GIS-based random forest classifier. *Bull. Eng. Geol. Environ.* **2019**, *78*, 2835–2849. [CrossRef]
38. Wang, Y.; Song, Q.; Du, Y.; Wang, J.; Zhou, J.; Du, Z.; Li, T. A random forest model to predict heatstroke occurrence for heatwave in China. *Sci. Total Environ.* **2019**, *650*, 3048–3053. [CrossRef]
39. Wang, Z.; Lai, C.; Chen, X.; Yang, B.; Zhao, S.; Bai, X. Flood hazard risk assessment model based on random forest. *J. Hydrol.* **2015**, *527*, 1130–1141. [CrossRef]
40. Deng, M.; Chen, J.; Huang, J.; Niu, W. Agricultural drought risk evaluation based on an optimized comprehensive index system. *Sustainability* **2018**, *10*, 3465. [CrossRef]
41. Alexander, D.L.J.; Tropsha, A.; Winkler, D.A. Beware of R2: Simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models. *J. Chem. Inf. Model.* **2015**, *55*, 1316–1322. [CrossRef]
42. Wang, W.; Lu, Y. Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model. In Proceedings of the IOP Conference Series: Materials Science and Engineering, Kazimierz Dolny, Poland, 21–23 November 2019.
43. Zikmund, W.G.; Babin, B.J.; Carr, J.C.; Adhikari, A.; Griffin, M. *Business Research Methods: A South Asian Perspective*, 8th ed.; Cengage Learning: Boston, MA, USA, 2013.
44. Kursa, M.B.; Rudnicki, W.R. Feature selection with the Boruta package. *J. Stat. Softw.* **2010**, *36*, 1–13. [CrossRef]
45. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
46. Breiman, L. *Out-of-Bag Estimation*; Citeseer: Berkeley, CA, USA, 1996.
47. Taylor, S.J.; Letham, B. Forecasting at scale. *Am. Stat.* **2018**, *72*, 37–45. [CrossRef]
48. Wu, Z.; Lin, H.; Li, J.; Jiang, Z.; Ma, T. Heat wave frequency variability over North America: Two distinct leading modes. *J. Geophys. Res. Atmos.* **2012**, *117*. [CrossRef]
49. Zhang, K.; Chen, Y.H.; Schwartz, J.D.; Rood, R.B.; O'Neill, M.S. Using forecast and observed weather data to assess performance of forecast products in identifying heat waves and estimating heat wave effects on mortality. *Environ. Health Perspect.* **2014**, *122*, 912–918. [CrossRef] [PubMed]
50. Lee, H.J.; Lee, W.S.; Yoo, J.H. Assessment of medium-range ensemble forecasts of heat waves. *Atmos. Sci. Lett.* **2016**, *17*, 19–25. [CrossRef]
51. Qi, X.; Yang, J. Extended-range prediction of a heat wave event over the Yangtze river valley: Role of intraseasonal signals. *Atmos. Ocean. Sci. Lett.* **2019**, *12*, 451–457. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

Article

NowDeepN: An Ensemble of Deep Learning Models for Weather Nowcasting Based on Radar Products' Values Prediction

Gabriela Czibula [1,*,†], Andrei Mihai [1,†] and Eugen Mihuleţ [2]

[1] Department of Computer Science, Babeş-Bolyai University, 400084 Cluj-Napoca, Romania; mihai.andrei@cs.ubbcluj.ro
[2] Romanian National Meteorological Administration, 013686 Bucharest, Romania; eugen.mihulet@meteoromania.ro
* Correspondence: gabis@cs.ubbcluj.ro; Tel.: +40-264-405327
† These authors contributed equally to this work.

Abstract: One of the hottest topics in today's meteorological research is weather nowcasting, which is the weather forecast for a short time period such as one to six hours. Radar is an important data source used by operational meteorologists for issuing nowcasting warnings. With the main goal of helping meteorologists in analysing radar data for issuing nowcasting warnings, we propose *NowDeepN*, a supervised learning based regression model which uses an ensemble of *deep artificial neural networks* for predicting the values for radar products at a certain time moment. The values predicted by *NowDeepN* may be used by meteorologists in estimating the future development of potential severe phenomena and would replace the time consuming process of extrapolating the radar echoes. *NowDeepN* is intended to be a proof of concept for the effectiveness of learning from radar data relevant patterns that would be useful for predicting future values for radar products based on their historical values. For assessing the performance of *NowDeepN*, a set of experiments on real radar data provided by the Romanian National Meteorological Administration is conducted. The impact of a *data cleaning* step introduced for correcting the erroneous radar products' values is investigated both from the computational and meteorological perspectives. The experimental results also indicate the relevance of the features considered in the supervised learning task, highlighting that the radar products' values at a certain geographical location at a time moment may be predicted from the products' values from a neighboring area of that location at previous time moments. An overall *Normalized Root Mean Squared Error* less than 4% was obtained for *NowDeepN* on the cleaned radar data. Compared to similar related work from the nowcasting literature, *NowDeepN* outperforms several approaches and this emphasizes the performance of our proposal.

Keywords: weather nowcasting; machine learning; deep neural networks; autoencoders; Principal Component Analysis

Citation: Czibula, G.; Mihai, A.; Mihuleţ, E. *NowDeepN*: An Ensemble of Deep Learning Models for Weather Nowcasting Based on Radar Products' Values Prediction. *Appl. Sci.* **2021**, *11*, 125. https://doi.org/10.3390/app11010125

Received: 28 October 2020
Accepted: 18 December 2020
Published: 24 December 2020

1. Introduction

Weather nowcasting [1,2] refers to short-time weather prediction, namely weather analysis and forecast for the next 0 to 6 h. Nowadays, the role of nowcasting in crisis management and risk prevention is increasing, as more and more severe weather events are expected [3]. Large volumes of meteorological data, including radar, satellite and weather stations' observations, are held by meteorological institutes and available for analysis. Radars and weather stations are constantly collecting real-time data, while data about cloud patterns, winds, temperature are continuously gathered by weather-focused satellites. Thus, there is large amount of meteorological related data available to be analyzed using machine learning (ML) based algorithms for improving the accuracy of short-term weather-prediction techniques.

The World Meteorological Organization (WMO) [4] mentions that "nowcasting plays an increasing role in crisis management and risk prevention, but its realization is a highly complex task", the highest difficulties being related to the small-scale nature of convective weather phenomena. This implies that the Numerical Weather Prediction approach (NWP) [5] is not feasible and that the forecast has to rely mainly on the extrapolation of known weather parameters. As meteorological institutes worldwide expect climate changes including extreme rain phenomena [3], there is an increasing need for accurate and early warning of severe weather events. Considering the increased number and intensity of severe meteorological phenomena, predicting them in due time to avoid disasters becomes highly demanding for meteorologists.

Mainly due to the extremely large volume of data that has to be analyzed in a short period of time, issuing a nowcasting warning is a complex and difficult task. For operative meteorologists it is difficult to issue nowcasting warnings, as there is a large volume of meteorological data (radar, satellite, or other weather stations' observations) that has to be analyzed in order to take an appropriate decision. Besides, given the stochastic and chaotic character of the atmosphere, the evolution of certain weather phenomena are difficult to predict by human experts. Thus, machine learning (ML) and *deep learning* [1,2] techniques are useful for assisting meteorologists in the decision-making process, offering solutions for nowcasting by learning relevant patterns from large amount of weather data. Most of the existing operational and semi-operational methods for nowcasting are using the extrapolation of radar data and algorithms mainly based on cell tracking. Existing nowcasting techniques use various data sources which may be relevant for accurate nowcasting, such as: meteorological data (radar, satellite, meteorological observations) and geographical data (elevation, exposure, vegetation, hydrological features, anthropic features).

In the current study we used real radar data provided by one of the WSR-98D weather radars [6] of the Romanian National Weather Administration. Given its capability to determine the location, size, direction and speed of water droplets, the weather radar is an essential tool used by meteorologists for nowcasting. For this type of forecast, fast decisions are imposed, so with the aim of facilitating the analysis in an operative environment, the data retrieved by the radar is supplied to the meteorologist in the form of coloured maps, which are easy to assess on a brief overview. On such a map, each pixel corresponds to a geographical location and its colour represents a certain value of the displayed product. In a short period of time, commonly under one minute, the meteorologist can locate potential dangerous storm cells, analyse their vertical structure, relative speed and direction, the dimension of hail, top of the clouds, and other relevant parameters. At this point, the operator should appreciate the current phase of the storm and by extrapolating those values in time, usually up to one hour, predicts the future development (intensity and area affected) of the storm. Although extrapolating the radar echoes is one of the main techniques used in nowcasting, it presents weaknesses in terms of processing times and precision, all related to the skills and experience of the meteorologist.

Most of the currently existing operational and semi-operational methods for nowcasting use the extrapolation of radar data and algorithms mainly based on cell tracking [7–9]. However, an important limitation of existing centroid cell-tracking algorithms lies in the detection of storms segments having irregular shapes or with variable wind speeds, resulting in identification and tracking errors.

With the goal of helping meteorologists in analysing radar data for issuing nowcasting warnings, we are introducing in this paper a supervised learning model *NowDeepN* based on an ensemble of deep neural network (DNN) regressors for predicting the values for radar products which may be used for weather nowcasting. As an additional goal, we aim to empirically validate the hypothesis that similar values for the radar products at a given time moment for a certain geographical region are encoded in similar neighborhoods of that region at previous time moments. The values predicted by *NowDeepN* for the radar products at certain time moments may be used by the meteorologist in estimating the future development of potential severe phenomena and thus *NowDeepN* would replace the time consuming process for the operational meteorologists of extrapolating the radar echoes.

As a proof of concept, $NowDeepN$ is proposed for learning to approximate a function between past values of the radar products extracted from radar observations and their future values. Experiments will be performed on real radar data provided by the Romanian National Meteorological Administration and collected on the Central Transylvania region. Our experimental goal is to obtain an empirical evidence that in both normal and severe weather conditions the values for a radar product at a given moment in a certain location are predictable from the values of the neighboring locations from previous time moments. If this stands, the study may be further improved and extended on a larger scale. In addition, we are proposing a *data cleaning* step useful for correcting the erroneous input radar data. To the best of our knowledge it is the first time an approach such as $NowDeepN$ is introduced in the nowcasting literature.

To summarize the contribution of this paper, our goal is to answer the following research questions:

RQ1 Are deep neural networks able to predict the values for a radar product at a given moment in a certain geographical location from the values of its neighboring locations from previous time moments? To what extent this holds both for both normal and severe weather conditions? For answering this research question, we are introducing a supervised learning model $NowDeepN$ consisting of an ensemble of DNN-based regressors.

RQ2 How does the data cleaning step introduced for correcting the erroneous input data impact the predictive performance of the $NowDeepN$ model? In addition, to what extent is the proposed data cleaning step correlated with the meteorological perspective?

RQ3 How relevant are the features considered in the supervised learning task? More specifically, are the radar products' values from the neighboring area of a certain geographical location l at time $t-1$ relevant for predicting the radar products' values on location l at time t?

The remainder of the paper is structured as follows. The problem of weather nowcasting, its importance and main challenges are discussed in Section 2. Section 3 discusses about the fundamental concepts regarding the used machine learning models, whilst a literature review on supervised learning based weather nowcasting is presented in Section 4. Section 5 introduces our data model and the $NowDeepN$ hybrid model for predicting the values for radar meteorological products using deep neural networks. Section 6 presents the experimental setting and results, while an analysis of the obtained results and their comparison to related work is provided in Section 7. The conclusions of our paper and future improvements and enhancements of $NowDeepN$ are outlined in Section 8.

2. Weather Nowcasting

The World Meteorological Organization (WMO) [4] mentions that "nowcasting plays an increasing role in crisis management and risk prevention, but its realization is a highly complex task", the highest difficulties being related to the small-scale nature of convective weather phenomena. This implies that the Numerical Weather Prediction approach is not feasible and that the forecast has to rely mainly on the extrapolation of known weather parameters. In this context, there is a high need for automated tools and systems supporting the nowcasting meteorologist, so significant research and development have been carried out on the topic of nowcasting. In spite of those efforts, both the operative personnel in nowcasting and the beneficiaries (population, relevant public institutions) are in demand of even more efficient products and services concerning the weather forecast itself and the alerting system.

Most of the currently existing operational and semi-operational methods for nowcasting are using the extrapolation of radar data and algorithms mainly based on cell tracking. Dixon and Wiener developed a real-time cell tracker named TITAN [10] which is useful for single cells. A centroid method is used in Reference [10] to identify storm entities in consecutive radar scans and estimate future movement of the storm centroid by minimizing a cost function. SCIT was proposed by Johnson et al. [11] as a cell-tracking algorithm with a more more complex cell detection method. SCIT used reflectivity thresholds and a distance function between cells for tracking and estimating future positions. An operational nowcasting tool called TRT was developed by Hering et al. [7] and used by Meteo Swiss. TRT is a centroid cell-tracking method using radar data and, in addition to previously mentioned methods, a multiple sensor system. Jung and Lee [12] introduced a cell-tracking algorithm using fuzzy logic based on a large set of historic radar data, with the goal of minimizing errors induced by single features. Germany's National Meteorological Service DWD uses a nowcasting system called NowCastMIX developed by James et al. [8]. NowCastMIX employes remote and ground observations analyzed using fuzzy logic rules. AROME [13] is a small scale numerical prediction model, which is used by Meteo-France since 2008. Measurements used by AROME include measurements of the precipitation systems, of low level humidity, low-level wind, upper level wind and temperature and it provides a forecast for up to 30 h. AROME-NWC [9] was developed in 2015 on top of AROME, for nowcasting in the range of 0–6 h. The INCA system [14] was developed specifically for mountainous regions with the goal of forecasting for precipitation amounts, temperature, wind and convective parameters.

An important limitation of existing centroid cell-tracking algorithms lies in the detection of storms segments having irregular shapes or with variable wind speeds, resulting in identification and tracking errors. Errors also occur when storm cells are clustered together and the algorithm detects them as a single larger cell. In other cases, a single storm cell is detected as two or more cells at the same horizontal location but on different altitude levels. Given the high spatial and temporal resolution, NowCastMIX [8] estimates of severe storms change quickly with the assimilation of new observations, leading to difficulties in reasoning for meteorologists. It is also prone to an overestimation of the likelihood of severe convection. According to Reference [14], the INCA system provides high accuracy for temperature but comparatively low for wind and precipitation given the relatively low distribution of relevant stations in mountainous regions. The systems described in References [8,9,14], which are considered to be some of the most performant automated nowcasting systems at the present time are highly customized for their respective location and infrastructure, thus their performance and adaptability to other contexts is unclear.

Several industrial players have also shown an interest in the problem of weather forecasting and have developed various solutions in this direction: IBM Deep Thunder [15] aims to provide high-resolution weather prediction for a variety of applications, Panasonic Global 4D Weather [16] is a weather forecasting platform that uses Panasonic patented atmospheric TADMAR sensor collected data, ClimaCell company [17] offers weather forecasting solutions tailored for various weather-sensitive industries while TempoQuest [18] is a company putting on the market a proprietary forecasting software for commercial users and government agencies.

3. Machine Learning Models Used

This section reviews the fundamental concepts regarding the machine learning models used in this paper: deep neural networks, *autoencoders* and *t-Distributed Stochastic Neighbor Embedding*.

Supervised learning is a subfield of machine learning, dealing with the task of approximating a mapping from some input domain to some output domain based on input-output example pairs. The data set consisting of the pre-existing examples of input-output pairs is called the *training data set*. A *supervised learning algorithm* generalizes the training data, producing a function that, given an input outside of the training data set, can return a close enough approximation of the correct output. What can be considered a "good enough approximation" is dependent on the specific problem. The vast quantity of data available nowadays as well as the increasing power of computation make the supervised learning methods an extremely useful tool that can be used in a wide range of domains.

3.1. Deep Neural Networks

Neural network learning methods provide a robust approach to approximating real-valued, discrete-valued or vector-valued target functions [19]. As a biological motivation, neural networks have been modeled to be similar to learning systems that we can find in humans and animals, namely complex networks of neurons. This morphology has been adopted in computer science, by building densely interconnected systems that have as building blocks basic units, that take as input a series of real-valued numbers and produce a single real-valued output [19]. These basic units are called artificial neurons. Neural networks are suited for problems that deal with noisy, complex data, such as camera, microphone or sensor data. Their success is due to their similarity to effective biological systems, that are able to generalize and associate data that has not been explicitly trained upon during the training phase, and correlate that data to a class where it belongs. Each neuron of the network has an array of parameters, based on which it processes the input data, called weights. The weights are adjusted during the training phase, based on the error of the network. The error represents the difference between the correct output and the network output. The learning algorithm used for adjusting the weights based on the error is the *backpropagation* algorithm.

Unlike classical neural networks, deep neural networks (DNNs) [20] contain multiple hidden layers and have a large number of parameters which makes them able to express complicated target functions, that is, complex mappings between their input and outputs [21]. Nowadays, DNNs are powerful models in the machine learning literature applied for complex classification and regression problems from various domains. Machine learning models, including deep ones, have been successfully applied for developing forecasting models such as for bank failures [22], prediction markets [23] or gambling [24]. Due to their complexity, large networks are slow to use and are prone to *overfitting*, which is a serious problem in DNNs. Overfitting is a major problem for supervised learning models, in which the model learns “by heart” the training data, but it does not have the capability to generalize well on the testing data. An overfit model is discovered through a very good performance on the training data, but a much lower performance on the testing set. A possible cause for overfitting in DNNs is the limited training data, as in such cases the relationships learned by the networks may be the result of sampling noise. Thus, these complex relationships will exist in the training data but not in real test data [21]. There are various methods for addressing overfitting and reducing it, such as—(1) stopping the training when the performance on a validation set starts decreasing; (2) introducing weight penalties through regularization techniques soft weight sharing [25]; (3) applying cross-validation; (4) extending the data set to include more training examples; and (5) *dropout* by randomly dropping neurons and their connections during training.

Regularization stands for an ensemble of techniques that have as purpose the simplification of the model, in order to avoid overfitting. Dropout is the regularization technique applied for neural networks. This process consists in deactivating some neurons during the training process, forcing the network to achieve the result by using a reduced (and simpler) neuron configuration. During prediction phase, the neurons will be reactivated. The selection of which neurons to keep active is done by a probability p, chosen arbitrarily, and dropped out by a probability of 1-p [21].

3.2. Autoencoders and Principal Component Analysis

Autoencoders (AEs) [20] are deep feed forward neural networks which aim to learn to reconstruct the input, being known in the machine learning literature as *self-supervised* learning systems. An AE has two main components: an *encoder* and a *decoder*. Assuming that the input space is $\mathbb{R}^n$, the encoder part learns a mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, while the decoder learns the function $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$. If $m < n$, the network learns to compress the input data into a lower dimensional latent space and to reconstruct it based on the latent representation, by learning the essential characteristics of the data. For avoiding the overfitting symptom (i.e., simply copying the input to the output), L1 regularization is applied on the encoded state and the model is called a *sparse* one. Autoencoders have been successfully applied in various tasks ranging from image analysis [26] and speech processing [27] to protein analysis and classification [28,29].

Principal Component Analysis (PCA) is a dimensionality reduction statistical technique heavily used in the ML field for tasks such as descriptive data analysis, data visualization or data preprocessing. It can be seen as an unsupervised learning technique that learns to represent the data in a new, lower-dimensional, space. Given a data set with n samples $(x_1, \ldots, x_n)$ and each sample having p attributes $(a_1, \ldots, a_p)$, the PCA algorithm will search for linear combinations of the variables (i.e., $\sum_{i=1}^{p} a_i \cdot c_i$ with $c_1, \ldots, c_p$ constants) such that they are linearly uncorrelated and that the first linear combination has the largest possible variance, the second has the largest possible variance while being orthogonal on the first linear combination and so on. These linear combinations are called *principal components*. The principal components can be found by computing the eigenvectors and eigenvalues of the covariance matrix of the data set, where the constants of each linear combination are given by the eigenvectors and the first linear combination is the one given by the eigenvector associated with the biggest eigenvalue, and so on [30].

4. Literature Review on Supervised Learning Based Weather Nowcasting

The literature contains various machine learning-based approaches for weather nowcasting. Relevant results obtained recently in predicting short-term weather are summarized in the following and the limitations of the existing solutions are emphasized.

Han et al. [31] use Support Vector Machines (SVM) are trained on box-based features in order to classify whether or not a radar echo $>$35 dBZ will appear on the radar within 30 min. The approach uses both temporal and spatial features, derived from vertical wind, and perturbation temperature. Greedy feature selection was used in order to finally select these features. The obtained results were around 0.61 Probability Of Detection (POD), 0.52 False Alarm Ratio (FAR), 0.36 Critical Success Index (CSI), which outperformed the Logistic Regression, J48, Adaboost and Maxent approaches compared against on the same data set.

Beusch et al. [32] present the COALITION-3 ("Context and Scale Oriented Thunderstorm Satellite Predictors Development") algorithm developed by MeteoSwiss. Its goal is to identify, track and nowcast the position and development of storms in a robust manner. In order to do this, it employs optical flow methods in combination with winds predicted by COSMO. In order to separate the movement of the storm from its temporal evolution, Lagrangian translation is used. Following this, the algorithm estimates future intensification and quantifies the probability and severity of the different risks associated with thunderstorms. This information is extracted by applying machine learning techniques to a data archive containing observations from the MeteoSwiss dual polarized Doppler radar system and information from other available systems.

Shi et al. [33] introduce an extension of the Long-Short Term Memory Network (LSTM), named ConvLSTM. Through experiments done on Moving MNIST and Radar Echo Dataset, the authors proved their method suitable for spatiotemporal data, having all the perks which come with a simple LSTM, preserving spatiotemporal features due to the inherited convolutional structure. Besides the above mentioned aspect other advantages of ConvLSTM over a basic LSTM are: transitions input-to-state and state-to-state are made in a convolutional manner, deeper models can produce better results with a smaller number of parameters. Their proposed architecture contains 2 networks, an encoder composed of 2 ConvLSTM layers and a forecasting network containing the same number of ConvLSTM layers and an additional 1×1 convolutional layer to generate final predictions.

Kim et al. [34] have also proposed a ConvLSTM-based model for precipitation prediction, but they used three-dimensional and four-channel data unlike Shi et al. [33], who used three-dimensional and only one-channel data. The four channels correspond to four altitudes. The proposed model, called DeepRain, predicts, based on radar reflectivity data, the amount of rainfall on a large scale. The experimental evaluation has proved a decrease of root mean square error (RMSE) with 23% when compared to linear regression and also a superior performance as compared to a fully connected LSTM. For rainfall prediction a RMSE value of 11.31 has been obtained on the test set.

Heye et al. [35] present a practical solution for leveraging the precipitation nowcasting problem starting from how the data is stored and preprocessed, the deep learning model used and up to the necessary frameworks and hardware. They used the ConvLSTM with peepholes cells described by Shi et al. [33] in an architecture inspired from those used for machine translation. The model is composed of an encoder and a decoder each made out of four ConvLSTM layers. They experimented with both taking the last step of the encoder and using attention for transferring information to the decoder, with the former producing better results in terms of Probability of Detection and Critical Success Rate and worse for the False Alarm Rate. The decoder uses both the encoded actual reflectivity and the predicted reflectivity. For accelerating learning, during training the ground truths are fed back into the decoder, while for evaluation the prior predictions are used. Additionally they noticed that using as start symbol a tensor of ones which represents high reflectivity when data is scaled to [0, 1] improves Probability of Detection and Critical Success Rate.

Shi et al. [36] proposed a benchmark for the nowcasting prediction problem and a new model. The benchmark includes a new data set, 2 testing protocols and 2 measurements for training, Balanced Mean Squared Error and Balanced Mean Absolute Error. Those measurements were necessary due to an imbalancement between the small amount of times where potentially dangerous events occured and normal amount of rainfall. The proposed model, Trajectory Gated Recurrent Unit (TrajGRU), deals better than Convolutional GRU, with the representation of location variant relationships. TrajGRU allows aggregation of the state along a learned trajectories. The architecture consist of an encoder and a forecaster part, inserting between RNN layers downsampling or upsampling operations depending upon region of the architecture. Their proposal show to be flexible and superior than other methods. The experiments done on Moving MNIST and precipitation nowcasting HKO-7 dataset emphasize the ability of the model to capture the spatiotemporal correlation.

Narejo and Pasero [37] have proposed a hybrid model combining Deep Belief Networks (DBN) with Restricted Boltzmann Machine (RBM) to predict different parameters of weather—air temperature, pressure and relative humidity—at a local level (i.e., restricted to a particular geographical area). Initially, the RBMs are trained unsupervisedly. Subsequently, the already trained RBMs are stacked to create a DBN. The DBN is supervisedly trained so as to predict the weather parameters.

Sprenger et al. [38] approached foehn prediction using AdaBoost machine learning algorithm. The authors motivate the choice of the model through the reasonableness of the balance between the predictive power, the computational speed and the interpretability of the results. Being trained with three years of hourly simulations data and using a modified decision stumps as weaker learners, the model achieved, on a validation data set, 0.88 sensitivity (or probability of detection) and 0.29 probability of false detection.

Yan Ji [39] approached the problem of short-term precipitation prediction from radar observations, using *artificial neural networks*. The nowcasting of the rain intensity was carried out on radar raw data and rain gauge data collected from China from 2010 to 2012. The reflectivity values were extracted from the raw data and were interpolated a 3D rectangular lattice grid of 1 km × 1 km in horizontal direction at the height of 1.5 and 3 km [39]. The collected data set was afterwards used for training the predictive model. A *root mean squared error* less than 5 (a minimum of 0.97 and a maximum of 4.7) was obtained. The experiments have shown a correlation coefficient R in the radar-rainfall estimation more than 0.6 and indicate that the accuracy rate of 36mins forecast is higher than 50% [39].

In a recent approach, Tran and Song [40] proposed a new loss function for the convolutional neural network (CNN) based models used for weather nowcasting. Using a computer vision perspective, the authors used Image Quality Assessment Metrics as loss functions in training, finding that the Structural Similarity function performed better than both MSE and MAE, especially by improving the quality of the images (i.e., the output images were much less blurry). However, the best performance was achieved by combining the Structural Similarity with the MAE and MSE loss functions.

Han et al. [41] proposed a CNN model for convective storms nowcasting based on radar data. The proposed model predicted whether radar echo values will be higher than 35 dBZ in 30 min, thus modelling the problem as a classification problem. The authors modeled the input radar data as multi-channel 3D images and proposed a CNN model that performs cross-channel 3-D convolution. In their model, the output is also an 3D-image, where each point of the image is a 0 if radar echo is predicted to be $\leq$35 dBZ in 30 min and 1 otherwise. A Critical Success Index (CSI) score of 0.44 was obtained.

Yan et al. [42] proposed a model for precipitation nowcasting employing a convolutional architecture using multihead attention and residual connections (MAR-CNN). The data fed into the network consists of radar reflectivity images on three elevation levels and other numerical features, such as cloud movement speed. In order to deal with the unbalanced classes of precipitations, extreme meteorological events have been oversampled. The proposed model outperformed several deep learning baselines obtained by using only several of the MAR-CNN components—a dual channel convolutional attention module, a dual channel convolutional model, a single channel convolutional model, as well as a GBDT and an SVM.

Limitations of Existing Approaches

Han et al. [31] highlight that support vector machines require feature reductions in order to be more accurate, so they might not be able to make use of all the information that a data set could provide. They also argue for the importance of collecting more data in order to train the machine learning models better. Another concern is that of feature selection: while other informative features do exist in the literature, it is not always straightforward to get them to a form usable for machine learning.

As most works [31,32] show, each system is mostly tested with some form of local real world data. This does not allow to accurately extrapolate a system's ability to adapt to other locations and their specific meteorological events and trends.

A general concern which later on can morph into a limitation is the possible data imbalance. The performance of the proposed methods can be diminished by the small number of risky weather conditions compared with normal rain [35,36]. The results from Reference [33] might not reflect the overall competence of ConvLSTM due to the fact that Shi et al. used a rather small data set for testing and and a low threshold for rain-rate. Another limitation mentioned [35] is the fact that predictions tend to lower values of precipitations in time due to previously less confident predictions being fed back into the recurrent network. The architecture proposed by Reference [35] is limited to the train data and does not learn from new data. A concern experienced by others when using the proposed TrajGRU cell [36] is the speed. The implementation of this operator turns out to be slower than a simple ConvGRU.

A common limitation of the solutions proposed by Reference [34] and Reference [37] is compromising the interpretability of the prediction results. The boosting-based solution [38] alleviate the issue of interpretability, but Dietterich has highlighted in one of his studies [43] the sensitivity to outliers as a particular disadvantage of boosting. Outliers can negatively affect the final predictions since they might be excessively weighted during the boosting steps. An additional limitation of most of the existing solutions is that they do not combine multiple data sources thus being deprived by an expected enhancement of the predictive capability [44].

5. Methodology

We further introduce our *NowDeepN* approach for weather nowcasting using *deep neural networks*. We start by describing in Section 5.1 the raw radar data used in our experiments. Section 5.2 introduces a theoretical model on which *NowDeepN* is based on, then Section 5.3 presents our proposal. The section ends with the evaluation measures we will use for assessing the predictive performance of $NowDeepN$.

5.1. Radar Data

The experiments in the current study were conducted on real radar data provided by one of the WSR-98D weather radars [6] of the Romanian National Weather Administration. The WSR-98D is a type of doppler radar used by meteorological administrations for weather surveillance, capable of remote detection of water droplets in the atmosphere, that is, clouds and precipitations, retrieving data on their location, size and motion. The WSR-98D scans the volume of air above an area over 70,000 square kilometers, and about every 6 min a complete set of about 30 base and derived products for 7 different elevations is being collected. The base products are *particle reflectivity* (R), providing information on particle location, size and type, and *particle velocity* (V), supplying information on particle motion, that is, direction and speed relative to the radar. Both products are available for several elevation angles of the radar antenna, and for each time step a set of seven data products, $R01$–$R07$ and $V01$–$V07$, is delivered, each of them corresponding to a certain tilt of the antenna. Among the derived products, of particular interest for this study is *VIL* (vertically integrated liquid), an estimation of the total mass of precipitation above a certain unit of area. The data in the NEXRAD Level III files is stored in a gridded format, each point of the grid corresponding to a geographical location and containing the value of a certain product at the respective time frame. In the data grid provided by the WSR-98D radar, the OX axis contains the longitude values, while the OY axis contains the latitude values.

5.2. Data Model

As shown in Section 1, the raw data provided by the radar scans during one day (24 h) on a certain geographic region was exported in the form of a sequence of matrices of $m \times n$ dimensional matrices, one matrix corresponding to a certain time moment t and a certain meteorological product p (i.e., each element from the matrix represents the value for the product p at a certain location from the map). As a set $Prod$ of multiple meteorological products are provided by the radar, the radar data collected at a time t may be visualized as a 3D data grid in which the OZ axis corresponds to the radar products.

For instance, Figure 1 depicts a sample 3D grid with $m = 2$ rows, $n = 2$ columns and three products ($Prod = \{R01, R02, R03\}$) recorded at a certain time stamp t. In the figure, the values for $R01$ are in the front matrix, $R02$ values are in the middle one and $R03$ in the matrix behind.

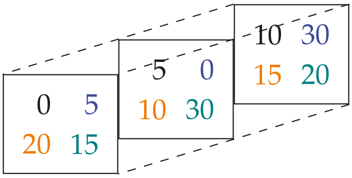
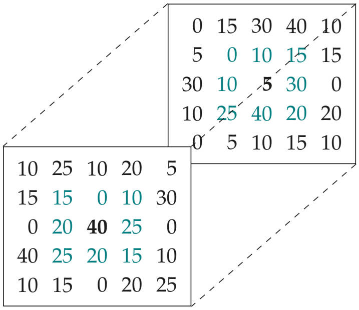


Figure 1. A sample 3D data grid.

During one day, a sequence of 3D data grids (as shown in Figure 1) corresponding to various time stamps is provided by the radar. Assuming that the radar records data every 6 min, 240 3D data grids are provided. For a certain location (i,j) on the map, a time moment t and a set $Prod$ of radar products, we are denoting by $V_t(i, j, l, Prod)$ the vector representing the linearized 3D data subgrid containing the radar products' values for a neighboring area of a certain length l surrounding the point (i,j), at time moment t.

As an example, let us consider a 5×5 dimensional data grid, the set $Prod = \{R01, R02\}$ of radar products, a time stamp t, the location (3, 3) ($i = 3, j = 3$) and a length l of 3 for the neighboring subgrid. Figure 2 depicts the 3D data grid containing values for $R01$ (front) and $R02$ (behind) for each cell from the data grid surrounding the point (3, 3), at time stamp

t. The point of interest as well as the 3D data subgrid of length 3 surrounding the point (3, 3) are highlighted.



0	15	30	40	10
5	0	10	15	15
30	10	**5**	30	0
10	25	40	20	20
0	5	10	15	10

10	25	10	20	5
15	15	0	10	30
0	20	**40**	25	0
40	25	20	15	10
10	15	0	20	25

Figure 2. The 3D data grid at time stamp.

The linearized 3D data subgrid (highlighted in Figure 2) is the 18-dimensional vector $V_t(3,3,3,Prod) = (15,0,0,10,10,15,20,10,40,5,25,30,25,25,20,40,15,20)$.

5.3. *NowDeepN* Approach

The regression problem we are focusing on is the following—to predict a sequence of values for a set *Prod* of radar products at a given time moment *t* on a certain location (*i*,*j*) on the map, considering the values for the neighboring locations of (*i*,*j*) at time moment $t{-}1$. *NowDeepN* approach consists of three main stages which will be further detailed: *data collection and cleaning*, *training* (building the learning model) and *testing*. *NowDeepN* uses an ensemble of DNNs for learning to predict the values of the radar products from the set *Prod* based on their historical values. The ensemble consists of *np* DNNs ($np = |Prod|$), one DNN for each radar product. We started from the intuition that using one network for each product would be more effective than using only one network for predicting all *np* values, as the mapping learned by the model should be specific to each radar product. Thus, we consider that the effectiveness of the learning process will be increased by using a DNN for each radar product and this will be empirically sustained by the experimental results (Section 6).

5.3.1. Data Collection and Cleaning

We mention that in the data set used in our case study, values for *R* and *V* products are available for only six elevations (i.e., *R*01–*R*04 and *R*06–*R*07, *V*01–*V*04 and *V*06–*V*07). The other three elevations delivered by the radar are missing since they are not regularly used in operational services, thus they are not stored in the same format as the rest of the elevations. The data gathered by the radar and exported as shown in Section 5.1 contains a special value that represents "No Data". This value is usually represented by −999 but we decided to replace it with 0 as in most cases this value refers to air particles with 0 reflectivity (i.e., no significant water droplets). "No data" may also represent air volumes which have returned no signal, for example if a sector with high reflectivity is between the radar and the respective location. In this case, replacing it with 0 is also correct, since the entire region is obturated and the data is not relevant for the learning process [45]. The radar data is also prone to different type of errors, meteorological and technical, which implicitly are to be found in the output data matrix. Meteorological errors (e.g., the underestimation of a particle's reflectivity) are difficult to identify and eliminate, but some errors occurring during the data conversion have been identified for *V*. For instance, the product *V* should only contain values from −33 to 33 but we found *invalid* values which are outside the range [−33, 33], such as −100. From a meteorological point of view, those erroneous values correspond to radar uncertainties in evaluating the direction and/or the speed of the particle, and are not taken into account in operative service since they are punctiform values and are irrelevant to the characteristics of a region.

We note that the cleaning process which is further described will be applied for the V values at each degree of elevation (i.e., V01–V04 and V06–V07). Thus, when we are using V in the following, we refer to the V value at a certain degree of elevation.

For reducing the noise that the invalid values of V represent, a *data cleaning* step is proposed. The underlying idea behind the cleaning step is to replace the invalid values of V on a certain point (i, j) with the weighted average of the valid V values from a neighborhood of length 13 surrounding the point. The weight associated to a certain neighbor of the point is inverse proportional to the Euclidian distance between the neighbor and the point, such that the closest neighbors' values have more importance in estimating the value of point. The reason for this cleaning step is that, from a meteorological viewpoint, V determines the direction and speed of air volumes, thus indicating neighbours that are more relevant for future value of points. The length 13 surrounding the point represents about 5 km in the physical world, a distance which commonly determines small gradients of the meteorological parameters.

Let us consider that (i, j) is the point having an erroneous value for V (e.g., -100) and this value has to be replaced with an approximation of its real value. We are denoting by $V(x,y)$ the value of the product V for the point (x,y) and by $\mathcal{N}_l(i,j)$ the 2D data subgrid representing the neighborhood of length l surrounding (i, j). For instance, the neighborhood $\mathcal{N}_3(3,3)$ of length 3 surrounding the point (3, 3) from the data matrix from Figure 3 is depicted in Figure 4.

0	15	30	40	10
5	−66	10	30	15
30	10	−100	15	0
10	25	−100	20	20
0	5	10	15	10

Figure 3. The sample 2D data grid. The values for the product V are displayed.

−66	10	30
10	−100	15
25	−100	20

Figure 4. The 2D data subgrid representing $\mathcal{N}_3(3,3)$.

For a certain point $(x,y) \in \mathcal{N}_l(i,j)$, $(x,y) \neq (i,j)$, we denote by $sim_{ij}(x,y) = \frac{1}{\sqrt{(x-i)^2+(y-j)^2}}$ the "similarity" between (i,j) and its neighbor (x,y). Certainly, the data points closer to (i,j) have a higher similarity degree and their value is more relevant in the cleaning process. Thus, the value $V(i,j)$ will be approximated with the weighted average of its valid neighbors, as shown in Formula (1)

$$V(i,j) = \sum_{\substack{(x,y)\in\mathcal{N}_l(i,j)\\(x,y)\ valid}} (w_{ij}(x,y) \cdot V(x,y)), \tag{1}$$

where $w_{ij}(x,y)$ represent the weight of point (x, y) and is computed as shown in Formula (2) by normalizing the similarity values $sim_{ij}(x,y)$ such that $\sum\limits_{\substack{(x,y)\in\mathcal{N}_l(i,j)\\(x,y) valid}} w_{ij}(x,y) = 1$. We note that this normalization assures that the approximated V values represent valid ones, that is, ranging in the interval [−33, 33].

$$w_{ij}(x,y) = \frac{sim_{ij}(x,y)}{\sum\limits_{\substack{(x',y')\in\mathcal{N}_l(i,j)\\(x',y')\ valid}} sim_{ij}(x',y')}. \tag{2}$$

As previously mentioned, in the cleaning process we considered a length $l = 13$ for the neighborhood. But, if all data points from the neighborhood of length l ($\mathcal{N}_l(i,j)$) are invalid, we are incrementally increasing l until the neighboring area will contain at least one valid point. In this case, Formula (1) is applied again using the new length l for estimating the value $V(i,j)$.

After the data was cleaned as previously shown, the data set is prepared for further training the *NowDeepN* regressor, using the representation described in Section 5.2. We denote, in the following, by $Prod = \{p_1, p_2, \ldots, p_{np}\}$ the set of radar products we are using in our approach. The radar data set cleaned as previously described is split into np subsets D_k, $1 \le k \le np$, a data set corresponding to each radar product. A DNN will be afterwards trained on each D_k, for learning to predict the value of the radar product p_k at a time moment t on a certain geographical location l, based on the radar products' values from the neighborhood of l at time $t-1$.

A training example from a data set D_k, $1 \le k \le np$ is in the form $< x_k, y_k >$, where:

- an instance x_k is the vector $V_{t-1}(i,j,l,Prod)$ (see Section 5.2) representing the linearized 3D data grid expressing the neighborhood of length l surrounding a point (i,j) at a time moment $t-1$;
- the label y_k of the instance x is the value for the radar product p_k for the location (i,j) from the map and time moment t (the time moment following the timestamp corresponding to instance x).

Each data set D_k will contain, for each data point from the analyzed map, examples in the form $< x_k, y_k >$ (as previously described). As a preprocessing step before training, the data sets D_k are normalized to [0,1], using the *min-max* normalization.

5.3.2. Building the *NowDeepN* Model

Using the data modelling proposed in Section 5.2 and previously described, we aim to build a supervised learning model *NowdDeepN* consisting of an ensemble of DNNs for expressing np functions (hypotheses) h_k, $1 \le k \le np$ such that $h(x_k) \approx y_k$.

One of the difficulties regarding the regression problem previously formulated is that the training data sets D_k built as shown in Section 5.3.1 are highly *imbalanced*. More specifically, there are a lot of training instances labeled with zero (i.e., $y_k = 0$) corresponding to points on the map without specific weather events and a much smaller number of instances with a non-zero label (i.e., corresponding to a severe meteorological phenomenon). The imbalanced nature of the data may lead to a regressor which is biased to predict zero values, as the majority of the training examples used for building the regressor were zero-labeled. A number of np DNN regressors N_k, $1 \le k \le np$ will be trained on the data sets D_k, such that the model M_k will learn to provide estimates for the radar product p_k.

Each of these DNN regressors outputs a single value which represents the prediction for the next time step for that regressor's product. The output value is given by an linear activation function, while the hidden neurons use the ReLU activation function [46]. The loss we used was the mean squared error and the optimizer used was Adam [47]. For regularization we used one drop-out layer with the default Keras parameters.

5.3.3. Testing

For assessing the performance of $NowDeepN$, a *cross-validation* testing methodology is applied on each of the data sets D_k. The data sets D_k are randomly splitted in 5 folds. Subsequently, 4 folds will be used for training and the remaining fold for testing and this is repeated for each fold (5 times).

For each training-testing split, two evaluation measures are used and computed for each training-testing split: *Root mean squared error* (RMSE) and *Normalized root mean squared error* (NRMSE) [48]. The RMSE computes the square root of the average of squared errors obtained for the testing instances. The NRMSE represents the normalized RMSE, obtained by dividing the RMSE value to the range of the output and is usually expressed as a percentage. The regression related literature indicates NRMSE as a good measure for estimating the predictive performance of a regressor. Lower values for RMSE and NRMSE (closer to zero) indicate better regressors. For a more precise evaluation of the results, the values for the evaluation measures (RMSE and NRMSE) are also computed for the non zero-labeled instances ($\text{RMSE}_{non-zero}$, $\text{NRMSE}_{non-zero}$).

The RMSE and NRMSE values are computed for each data point from the grid (geographical area) and then are averaged over all grid points. As multiple experiments (training-testing data splits) are performed, the values for the evaluation measures were averaged over the 5 runs and a 95% *confidence interval* (CI) [49] of the mean value is computed.

6. Experimental Results

Experiments were performed by applying $NowDeepN$ model on real data sets provided by the Romanian National Meteorological Administration and collected on the Central Transylvania region, using the methodology introduced in Section 5.

6.1. Data Set

This study uses data provided by the WSR-98D weather radar [6] located in Bobohalma, Romania and stored in the NEXRAD Level III format, as described in Section 5.1. The day used as case study is the 5th of June 2017, a day with moderate atmospheric instability manifested through thunderstorms accompanied by heavy rain and medium-size hail. In our study we selected an area from the central Transylvania region (parts of Mureş, Cluj, Alba and Sibiu counties) representing a grid having the geographical coordinates (46.076 N, 46.725 N, 23.540 E and 25.064 E). In the chosen geographical area, there were two distinct episodes with intense meteorological events in 5 June 2017: the first one between approximately 09:00 and 11:00 UTC, and the second one between approximately 12:00 and 17:00 UTC, with the most severe events taking place between 14:00 and 15:00 UTC. Concerning these phenomena, the National Meteorological Administration issued five severe weather warnings, code yellow.

The data grid provided by the radar for the selected geographical area at a given time moment is fit to a matrix. The radar provides one data matrix for each radar product. As stated in Section 5.1, the radar data is split into multiple time stamps, each time stamp representing data gathered by the radar every 6 min (the radar takes 6 min to gather the data for the area). The radar data used in our case study has been recorded between 00:04:04 UTC and 23:54:02 UTC.

In the current study, we are using only 13 products (i.e., $np = 13$): base *reflectivity* (R) of particles on six elevations ($R01$–$R04$, $R06$–$R07$) *velocity* (V) on six elevation ($V01$–$V04$, $V06$–$V07$) and the estimated quantity of water (VIL) contained by a one square meter column of air. Thus, the set of considered radar products is $Prod = \{R01, R02, R03, R04, R06, R07, V01, V02, V03, V04, V06, V07, VIL\}$. Accordingly, $NowDeepN$ ensemble of regressors will predict 13 values, corresponding to the products previously enumerated. Our study uses only R, V and VIL products, as they are mostly used by meteorologists for weather nowcasting.

As mentioned in Section 5.2, we consider each point in the grid an instance. For each point we consider its neighbours in a certain radius. We decided to select a value of 13 for the length l of the neighborhood surrounding each point (see Section 5.3.1). More exactly, the 2D data subgrid representing the neighborhood for a point is a 13 by 13 matrix. The reason for choosing 13 as the dimensionality of the neighborhood is that it represents about 5 km in the physical world, which, from a meteorological view, is a common distance to determine small gradients. Thus for each instance we are considering a matrix of 13 by 13 points, each of which have 13 products. Therefore, for each instance we have 13 × 13 × 13 = 2197 attributes. For each timestamp we have a grid of the size 400 × 312. We only used the instances for which we could get the entire neighbourhood (i.e., where the neighbourhood matrix would not exceed the limit of the grid), thus obtaining 116.400 instances per timestamp. The day used as a case study contains 231 timestamps, thus of our data set consists of 26.888.400 instances. The data used in the experiments are publicly available at http://www.cs.ubbcluj.ro/~mihai.andrei/datasets/nowdeepn/.

6.2. Data Analysis

In order to estimate the impact of the data cleaning step, we analyzed the data set before and after cleaning. For each of the data products $V01$, $V02$, $V03$, $V04$, $V06$ and $V07$ (which were possibly cleaned) and each time stamp t, $1 \leq t \leq 231$, we computed the average values of the radar products for all the cells from the analyzed grid. Additionally, the average of the mean values for all V products and all the cells from the grid were calculated for each time stamp.

Figure 5 comparatively depicts the variation of the mean V value with respect to each time stamp (ranging from 1 to 231) for three cases: (1) *before the cleaning* step; (2) before cleaning but *ignoring the invalid values*; and (3) *after the cleaning*. A step of 10 was selected on OX axis (i.e., an hour). Graphical representations similar to the ones from Figure 5 were created for $V01$–$V07$, as well. The time series plots for $V01$ and $V06$ are illustrated in Figures 6 and 7, respectively.

Analyzing the plots from Figures 5–7 and comparing the evolution of values *before cleaning* (red coloured), before cleaning but ignoring *invalid values* (blue coloured) and *after cleaning* (green coloured) we observe the following. At lower degrees of elevations (see the plot for $V01$ from Figure 6) there are much more invalid values than at higher degrees of elevation (see the graph for V06 from Figure 7). Besides, from a meteorological viewpoint, higher degrees of elevation are related to higher altitudes, implying a less chaotic air circulation, leading to more precise radar soundings. The impact of the data cleaning step is noticeable from the time series plots, as for $V01$ the graph before data cleaning significantly differs from the graph after cleaning. The two graphs have very different shapes and this suggests that the noise introduced in data after the cleaning step is considerably smaller than the noise existing in data before replacing the invalid V values. We note that a similar situation has been observed for $V02$–$V04$ as well.

Much more, the time series plots before data cleaning but ignoring invalid values (blue coloured) resembles to the graph after the invalid values were cleaned (green coloured). Figure 8 depicts a zoomed-in version of the graphs from the upper side of Figure 6 (the time series before cleaning but ignoring the invalid values and time series after cleaning). At higher degrees of elevation (V06), there is no significant difference between the evolution of V values before and after cleaning (the red coloured and green coloured plots from Figure 7). The shapes of the two plots are very similar for $V06$ and this was also observed for $V07$.

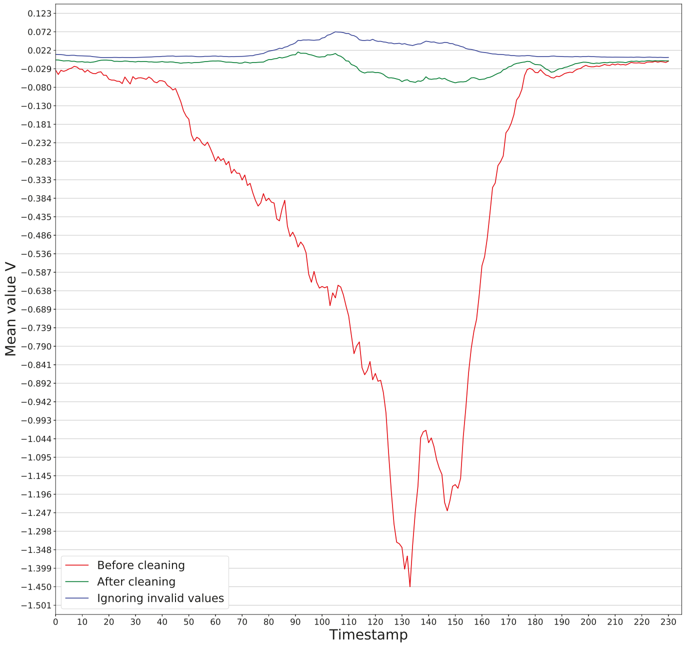


Figure 5. Time series plot for mean V values: ignoring invalid values, before and after cleaning.

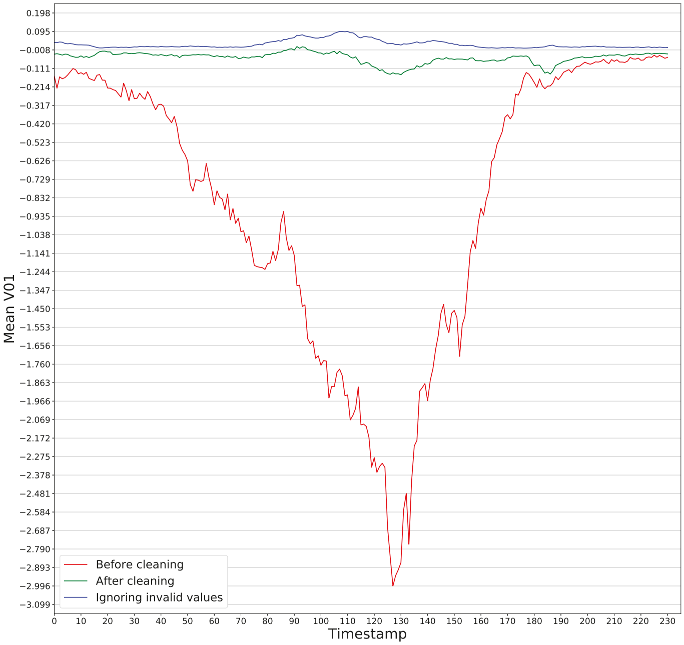


Figure 6. Time series plot for average $V01$ values: ignoring invalid values, before and after cleaning.

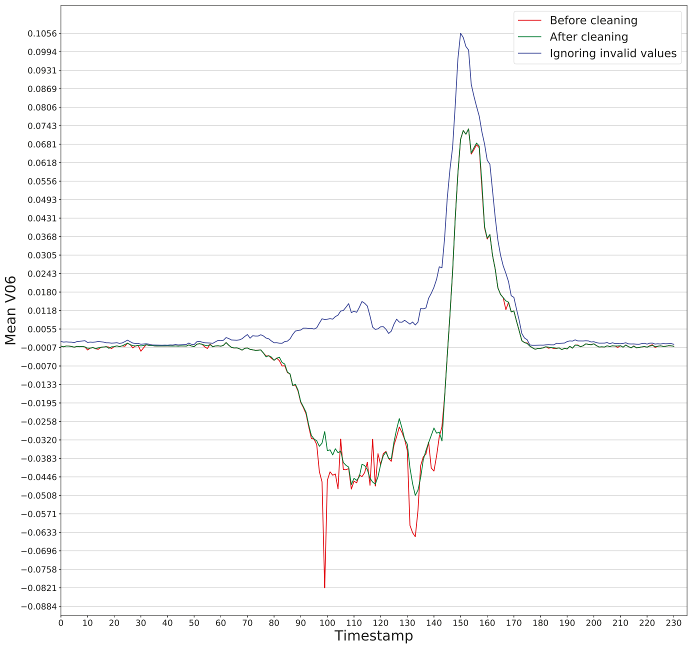


Figure 7. Time series plots for average $V06$ values: ignoring invalid values, before and after cleaning.

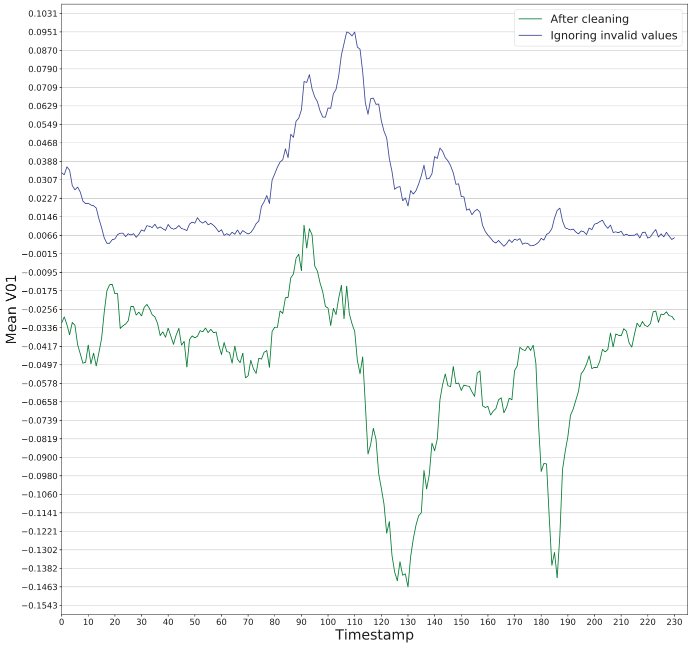


Figure 8. Time series plot for average $V01$ values: ignoring invalid values and after cleaning.

All the previous observations lead us to the hypothesis that the cleaning step would impact the overall performance of *NowdDeepN*, and this should be visible at least at lower degrees of elevations for *V*.

6.3. Results

This section presents the experimental results obtained by applying *NowDeepN* approach on the data set described in Section 6.1. For the DNNs used in our experiments, the implementation from the Keras deep learning API [50] using the Tensorflow neural networks framework was employed. The code is publicly available at Reference [51]. Given the fact that our data was quite high-dimensional, as mentioned in Section 6.1, we needed a relatively complex neural network. Each of The DNN regressors in the ensemble contains 12 hidden layer, with the following number of neurons: one layer with 200 neurons, one layer with 2000 neurons, 5 layers with 500 neurons and 5 layers with 100 neurons. These networks were trained for 30 epochs using 1024 instances in a training batch.

As stated at the beginning of the paper, we aim to answer research question RQ1 by assessing the ability of *NowDeepN* to predict the values for the radar products at a given moment in a certain geographical location from the values of its neighboring locations from previous time moments. Besides, we intend to analyze how correlated are our computational findings with the meteorological evidence. Table 1 depicts the obtained results together with their 95% CI. The columns of the table illustrate the evaluation measures computed for all 13 products (second column), the average values computed for all six *R* products (third column), the average values computed for all six *V* products (fourth column) as well as for *VIL* (fifth column). In order to allow an easier interpretation of the results from a meteorological perspective, we also illustrate in Table 1 the *Mean of Absolute Errors* (the average of the absolute errors obtained for the testing instances) for all instances (MAE), as well as only for the non-zero labeled instances (MAE$_{non-zero}$).

Table 1. Experimental results obtained using *NowDeepN*. 95% CI are depicted.

Evaluation Measure	All 13 Products	All *R* Products	All *V* Products	*VIL*
MAE	0.58 ± 0.02	0.76 ± 0.03	0.41 ± 0.02	0.53 ± 0.02
RMSE	2.25 ± 0.12	2.73 ± 0.17	1.44 ± 0.07	1.62 ± 0.10
NRMSE	3.27% ± 0.17%	3.91% ± 0.24%	2.15% ± 0.11%	2.32% ± 0.14%
MAE$_{non-zero}$	4.02 ± 0.12	5.51 ± 0.17	2.73 ± 0.12	2.89 ± 0.04
RMSE$_{non-zero}$	5.93 ± 0.14	7.63 ± 0.15	3.50 ± 0.15	3.9 ± 0.18
NRMSE$_{non-zero}$	8.60% ± 0.21%	10.91% ± 0.22%	5.22% ± 0.22%	5.63% ± 0.26%

From a meteorological point of view, the MAE for both all and non-zero instances is a satisfactory one, meaning that the predicted value is on the same level or on a neighbouring level on the product value scale.

The following two figures depict a comparison between the real data gathered by the radar (Figure 9) and the prediction made by *NowDeepN* (Figure 10). The timestamp of the comparison was chosen so that there was significant meteorological activity (14:37 UTC is in the middle of the meteorological event, as described in the data set used in Section 6.1). The figures depict product R01, chosen because it is one of the most relevant radar products for nowcasting, as it provides information on the precipitation and usually contains more non-zero data than upper levels. Figure 9 represents the real data collected by the radar, while Figure 10 represents the predicted data, given the real data at 14:31:15 UTC. The figures represent the (real or predicted) values of R01 *over a geographical area*. Each pixel in the images represents a geographical location of roughly 1 km^2. The values on axes OX and OY represent the coordinates *of a pixel* inside the image, relative to the (0,0) origin in the upper-left corner. While the values along the axes do not represent actual longitude and latitude values, the OY axis runs along the latitudes and OX axis runs along the longitudes (the (0,0) origin of the images being the most north-western point of the geographical area the image represents).

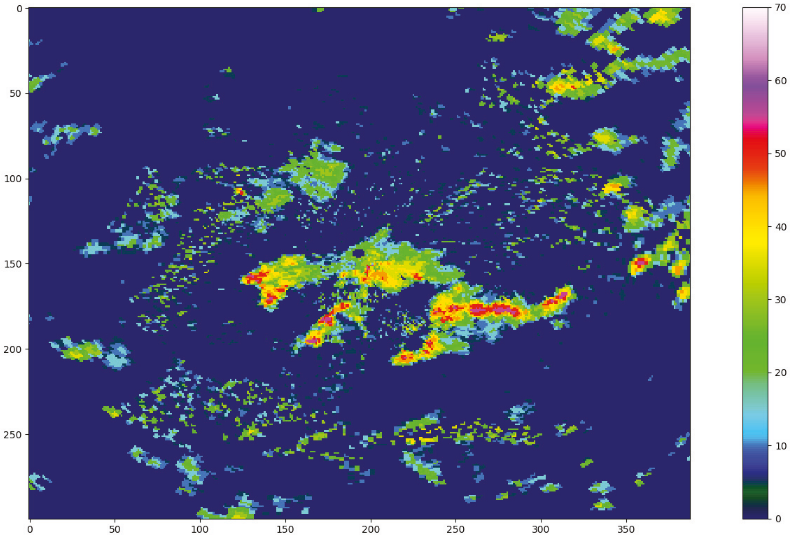


Figure 9. Real data for product *R*01 (reflectivity at the lowest elevation angle, measured in dBZ with values ranging form 0 to 70) at 14:37:22 UTC.

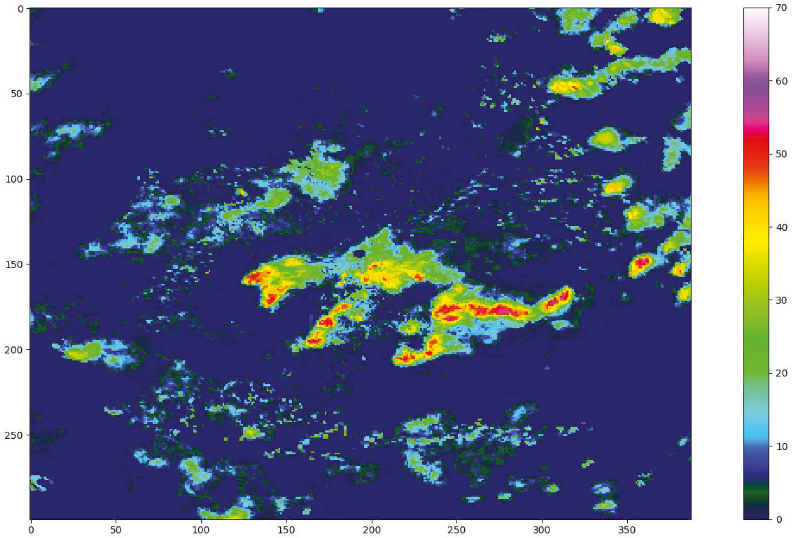


Figure 10. Predicted data for product *R*01 (reflectivity at the lowest elevation angle, measured in dBZ with values ranging form 0 to 70) at 14:37:22 UTC.

In Table 1 an average NRMSE of less than 4% is reported for the *R* products, which would entail a close resemblance between the predicted data and the real data, resemblance which can be observed in the comparison between the Figures 9 and 10. It can be observed that the predicted pattern closely follows the real pattern, closely matching the shape and intensity of R01 in the studied area. There are, however, visible limitations of the prediction. There is a clear smoothing effect present, very clearly seen around areas where there are scattered points with non-zero values, the prediction tends to smooth out the area between them, resulting in a much larger areas of close to 0 non-zero values (colored in dark green) than in the real data. This effect can be also seen on areas with points with higher values, for example, around the point at roughly (170, 230), where in the real data present some ragged shape of higher intensity points, but the predicted data would present the area as a smooth shape, much less ragged, the space between the higher intensity points being predicted with a similar intensity. Another example can be seen at (110, 120), where, in the real data, there is a small shape with extremely high values; but in the predicted data, while the real shape is largely retained, the intensity is greatly diminished. Still, these kinds of effects are on a small scale relative to the entire area represented by the figure. Smoothing may be responsible for the higher NRMSE obtained for non-zero values, as areas with higher values are more affected by smoothing than areas with zero values.

7. Discussion

In this section we are analysing the performance of *NowDeepN* approach in order to answer research questions RQ2 and RQ3. First, we are going to assess the impact of the data cleaning step on the performance of *NowDeepN* (Section 7.1) and to estimate the relevance of the manually engineered features used in the training process (Section 7.2). Then, we continue in Section 7.3 with comparing our results with similar results obtained in the literature.

7.1. Impact of the Data Cleaning Step

As previously shown in Section 6.1, the training data set contains instances with errors, particularly at lower degrees of elevation. Obviously, the noisy training data may affect the performance of the learning task. In this regard, a data cleaning step motivated by the meteorological perspective has been introduced in Section 5.3.1 for replacing the invalid values in the data set which were identified mostly for the product V at lower degrees of elevation ($V01$–$V04$). The analysis we performed on data after it was cleaned (Section 6.2) led us to the hypothesis that the cleaning step would impact the overall performance of *NowdDeepN*, and this should be visible at least at lower degrees of elevations for V. Intuitively, as the data cleaning is correlated with the meteorological evidence, we expect a better performance of *NowDeepN*, particularly at lower degrees of elevation.

In order to empirically validate the hypothesis that the cleaning step improves the predictive performance of *NowDeepN*, we have evaluated the model trained on the uncleaned data set, using the same methodology introduced in Section 5.

Table 2 depicts the results obtained applying *NowDeepN* on the uncleaned data. Comparing the results with those obtained on the cleaned data (Table 1) we observe an improvement in the predictive performance of *NowDeepN* achieved on the cleaned data. The last column from the Table 2 illustrates the improvement obtained using the cleaning step, computed for each evaluation measure E (i.e., RMSE, NRMSE, RMSE$_{non-zero}$, NRMSE$_{non-zero}$) for all 13 radar products. The improvement is computed as $\frac{E_{uncleaned}-E_{cleaned}}{E_{uncleaned}}$.

Figure 11 depicts a very similar situation as Figure 10, the only difference being that the image represents values for R01 predicted by *NowDeepN* trained on *uncleaned* data. Similar to Figures 9 and 10, the image in Figure 11 represents a geographical area with each pixel representing a geographical location of roughly 1 km^2. Again, the values on axes OX and OY represent the coordinates of pixels inside the image relative to the $(0,0)$ origin in the upper-left corner, while axis OX runs along the longitudes and axis OY runs along the latitudes, with the $(0,0)$ origin being the most north-western point in the geographical area represented by the image. The most erroneous values are on the V products, yet the errors can still greatly affect the other products such as R01, presented in the figure. While in the predicted data the shapes and intensities are largely retained as in the real data, some significant anomalies can be observed. For example, around the point at (140, 250) in the predicted data can be seen an area of higher valued points that is completely absent int the real data, with no means to attribute it to the smoothing effect. Also, in the upper right corner at around (10, 380), there can be seen in the real data a significant shape with quite high values at the interior that is simply removed in the predicted data, although the surrounding shapes are much better represented. There are other such anomalies that cannot be explained by the smoothing effect that appear in the data predicted by *NowDeepN* trained on uncleaned data. None of these anomalies appear on the data predicted *NowDeepN* trained on cleaned data. This suggests that there were some erroneous values of V in those areas that also affected how *NowDeepN* predicted the other products, and thus, cleaning the data yields much better results.

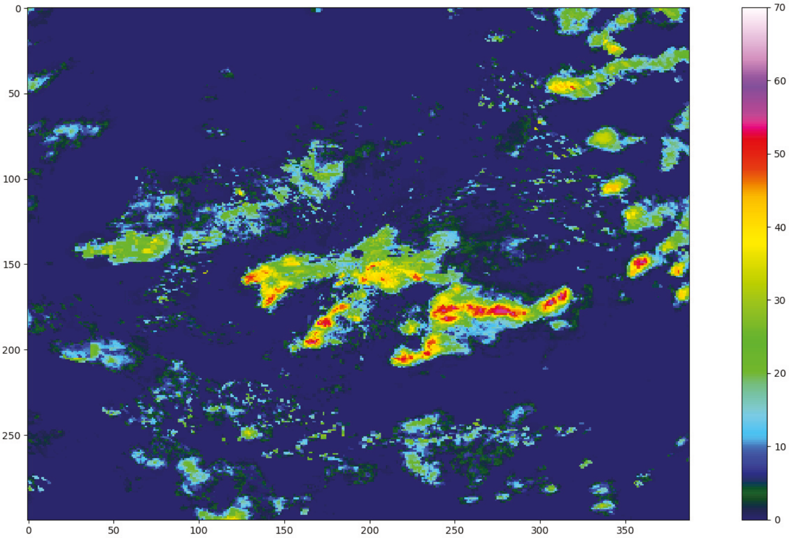


Figure 11. Predicted data, using the model trained on uncleaned data, for product *R*01 (reflectivity at the lowest elevation angle, measured in dBZ with values ranging form 0 to 70) at 14:37:22 UTC.

7.2. Relevance of the Used Features

We aim to further analyze the *NowDeepN* approach by determining the relevance of the features used in the training process. More precisely, our goal is to examine if the radar products' values from a neighborhood of a certain location at time t are suitable for predicting the products' values at time $t + 1$, for the same location. The analysis from this section is conducted for answering question RQ3.

For determining the significance of the features, we are comparing the results of *NowDeepN* using the original set of features with those obtained by applying *NowDeepN* after the prior application of a feature extraction step. Two feature extractors will be applied on the original set of features, for reducing the dimensionality of the input data.

Table 2. Experimental results obtained using *NowDeepN* on the uncleaned data. 95% CI are used for the results. The last column illustrates the improvement achieved applying *NowDeepN* on the cleaned data, considering all 13 radar products.

Evaluation Measure	All 13 Products	All *R* Products	All *V* Products	*VIL*	Improvement (%) (Cleaning Step)
RMSE	4.98 ± 0.06	4.97 ± 0.10	3.99 ± 0.07	5.24 ± 0.17	55%
NRMSE	7.27% ± 0.09%	7.10%± 0.15%	5.95% ± 0.10%	7.49% ± 0.24%	55%
$\text{RMSE}_{non-zero}$	10.05± 0.40	9.38 ± 0.23	10.88± 0.71	9.10 ± 0.31	41%
$\text{NRMSE}_{non-zero}$	14.68%± 0.59%	13.40%± 0.33%	13.24%± 1.06%	13.00% ± 0.44%	41%

1. A sparse denoising AE is applied on the original data using a dimensionality of 250 for the hidden state. The AE's latent space representation (i.e., the result of the encoding part) will be further used (as a lower-dimensional representation of the input data) by *NowDeepN* for the prediction task. For the AE implementation we have used the following model: 7 hidden layers, 3 hidden layers for encoding (1000, 750, 500 neurons respectively), 250 neurons on the encoding layer and 3 hidden layers for decoding (500, 750, 1000 neurons respectively), all of these layers using the ReLU activation function, with the exception of the encoding layer, which used the linear activation function; the output size is equal to the input size (2197 neurons), with the linear activation function; using mean squared error loss and the Adam optimizer; the network was trained for 30 epochs using a batch of 1024 instances.
2. The PCA algorithm is applied for a linear dimensionality reduction of the input data into a 250 dimensional space. For PCA we have used the existing scikit-learn Python implementation of the algorithm using 250 principal components and the default values for the other parameters [52].

Table 3 depicts the results obtained applying *NowDeepN* with a previous feature extraction step (AE/PCA). Comparing the results with those obtained without applying a feature extraction step (Table 1) we observe an improvement in the predictive performance of *NowDeepN* achieved without a prior feature extraction step. The last column from the table illustrate the improvement obtained by *NowDeepN* on the original data (computed as the difference between the measure on the original and on the reduced data divided to the value obtained on the reduced data).

The results depicted in the last column from Table 3 empirically demonstrate that the features (i.e., the radar products' values from a neighborhood of a certain location at time t) used in training *NowDeepN* are relevant for predicting the radar products' values at time $t + 1$, for the same location. The relevance of the features is validated by the fact that a dimensionality reduction technique (AE/PCA) applied prior to the classification using *NowDeepN* does not improve the learning performance. The last column from Table 3 also reveals that AEs preserve better than PCA the characteristics of the data when reducing its dimensionality, which is expectable as AEs perform a non-linear mapping whilst PCA a linear one.

7.3. Comparison to Related Work

The literature review from Section 4 revealed various approaches developed in the nowcasting literature using machine learning methods.

Table 3. Experimental results obtained using *NowDeepN* with a previous feature extraction step. 95% CI are used for the results. The last column illustrates the improvement achieved applying *NowDeepN* on the original data, considering all 13 radar products.

Feature Extraction	Evaluation Measure	All 13 Products	All *R* Products	All *V* Products	*VIL*	Improvement (%) without Feature Extraction
AE	RMSE	2.40 ± 0.07	2.93 ± 0.08	1.51 ± 0.05	1.76 ± 0.11	6%
	NRMSE	3.49% ± 0.10%	4.19% ± 0.12%	2.26% ± 0.08%	2.51% ± 0.16%	6%
	RMSE$_{non-zero}$	6.75± 0.29	8.81 ± 0.41	3.87 ± 0.14	4.55± 0.49	12%
	NRMSE$_{non-zero}$	9.79% ± 0.41%	12.58% ± 0.59%	5.78% ± 0.20%	6.49% ± 0.70%	12%
PCA	RMSE	2.76 ± 0.03	3.40 ±0.15	2.16 ±0.09	2.52 ±0.13	18.50%
	NRMSE	4.01% ±0.05%	4.86%± 0.22%	3.22%±0.14%	3.60% ±0.19%	18.47%
	RMSE$_{non-zero}$	7.56 ± 0.31	9.80± 0.79	5.58 ±0.19	5.94 ±0.37	21.55%
	NRMSE$_{non-zero}$	10.96% ±0.45%	14.01% ±1.13%	8.33%± 0.29%	8.49% ±0.53%	21.54%

We start the comparison between *NowDeepN* and related work by comparing our model to a simple baseline model, the *linear regression* (LR). For an exact comparison, the data model used for *NowDeepN* (Section 5.2) was used for the LR model as well. By applying the LR on the dataset described in Section 6.1, an overall RMSE for the non-zero values (RMSE$_{non-zero}$) of 6.094 was obtained. Surprisingly, there is only 3% improvement achieved by *NowDeeP* on our data, with a higher improvement on *V* (about 12%). From a meteorological point of view, the minor improvement in reflectivity nowcasting (compared to the LR model) is probably determined by the fact that the model is predicting the values of the radar products for one time step, and on the particular day used in this study the convective structures detected by the radar display a relatively slow evolution due to the light to moderate wind, thus no rapid modifications between two radar scans can be observed in terms of *R* value or location. Bigger improvements can be observed in *V* product values predictions, since the evolution of this product has a more stochastic character. Future work, dealing with prediction over more time steps, should display greater improvements compared to the benchmark LR model.

Even if there are numerous machine learning-based methods developed for nowcasting purposes, there are few methods focused on radar base products' values nowcasting, such as reflectivity nowcasting. Most of the related work focus on the precipitation nowcasting problem. We found four approaches having similar goal to our paper, that of predicting the future values of the radar products' values based on their historical values. The approaches from the literature which are the most similar to ours are those proposed by Yan Ji [39], Han et al. [31,41] and Yan et al. [42]. Even if the data sets used in the previously mentioned papers and the evaluation methodology differs from ours, we computed the evaluation measures reported in literature, trying to reproduce as accurately as possible the experiments from the related work.

The work of Ji [39] is focused on predicting only the reflectivity values which are further used for precipitation prediction. Experiments are performed only on radar data collected only for time stamps when storms occurred, disregarding the periods with normal weather (i.e., for which the *R* value is 0). Besides the minimum and the maximum values for the RMSE, the *Hit rate* (HR) is reported in Reference [39] as the percentage of instances for which the absolute error (between the predicted and the real value) is less than or equal to 5. For an accurate comparison with the work of Yan Ji [39], we trained our *NowDeepN* model only on the instances labeled with non-zero values for *R* and was tested only on non-zero instances. We also note that the evaluation from Reference [39] is performed only once, without using a cross-validation. Han et al. [31,41] focused on their works on predicting the *R* values (using SVM and CNN classifiers), more specifically if they exceed 35 dBZ. Considering that the positive class is the one for which the *R* values are larger than 35, the authors used three evaluation measures: (1) *critical success index* (CSI), computed as

$CSI = \frac{TP}{TP+FN+FP}$; (2) *probability of detection* (POD), $POD = \frac{TP}{TP+FN}$; and (3) *false alarm rate* (FAR), $FAR = \frac{FP}{FP+TP}$. The MAR-CNN model proposed for precipitation nowcasting by Yan et al. [42] used radar reflectivity images on three elevation levels and other numerical features and provided a RMSE of 7.90 for the predicted reflectivity values.

Table 4 summarizes the comparison between *NowDeepN* and the related work. The best values for the evaluation measures are highlighted.

Table 4. Comparison to the work of Yan Ji [39], Han et al. [31,41] and Yan et al. [42].

Model	RMSE	HR	CSI	POD	FAR
Our *NowDeepN*	5.34	0.75	**0.64**	**0.83**	**0.27**
ANN [39]	**[0.97, 4.7]**	**0.89**	–	–	–
CNN [41]	–	–	0.44	–	–
SVM [31]	–	–	0.36	0.61	0.52
MAR-CNN [42]	7.90	–	–	–	–

Table 4 reveals that *NowDeepN* outperforms the approaches proposed by Han et al. [31,41] in terms of CSI, POD and FAR evaluation measures. Overall, in 71% of the cases (5 out of 7 comparisons), the comparison is favorable to *NowDeepN*. Our proposal is outperformed only by the work of Yan Ji [39] which reported a better HR and a maximum RMSE slightly better than ours. This difference may occur due to the following: (1) the data sets used (both as training and and testing) are different and have particularities due to the geographic area (country) on which were collected (i.e., China [39] and Romania); (2) the testing data set from Reference [39] contains data collected on a relatively small area which may lead to a biased evaluation and an overestimated performance.

As previously mentioned, a lot of work has been carried out in the literature for precipitation nowcasting. Tran and Song [40] tackled the precipitation nowcasting problem from a computer vision perspective, by applying certain thresholds on the reflectivity values (5/20/40 dBZ). In order to measure the performance of *NowDeepN* (in terms of CSI, POD and FAR) for the aforementioned thresholds we transformed the predicted values to predicted classes, by denoting each predicted value lower or equal to a threshold as being in the negative class and each predicted value higher than the threshold as being in the positive class. We then applied the same transformation on the ground truth and computed the measures; following this process for each of the three thresholds (5/20/40 dBZ). Table 5 illustrates the comparison between *NowDeepN* and the model proposed by Tran and Song [40]. We mention that Tran and Song [40] provide for each evaluation measure ranges of values. Since an exact comparison cannot be provided (the datasets used for evaluation and the input data models are different) our comparison relies only on the magnitude of CSI, POD and FAR evaluation metrics. The best values obtained for each reflectivity threshold and evaluation metric are highlighted.

The comparative results from Table 5 highlight that *NowDeepN* obtained better results than the model proposed by Tran and Song [40] in 77.7% of the cases (7 out of 9 comparisons). We note the good performance of *NowDeepN* at higher values for the reflectivity threshold, which indicate the ability of our model to detect moderate and heavy precipitation and medium and large hail.

Table 5. Comparison to the work of Tran and Song [40].

Model	Reflectivity Threshold	CSI	POD	FAR
Our *NowDeepN*	5	**0.6475**	**0.8055**	**0.2326**
	20	**0.5386**	**0.7646**	**0.3543**
	40	**0.4290**	**0.6277**	**0.4245**
TrajGRU [40]	5	0.6729–0.7069	0.7646–0.8053	0.1579–0.1812
	20	0.2994–0.3208	0.3949 –0.4296	0.4443–0.4815
	40	0.0411–0.0549	0.0568–0.0859	0.7539–0.79

8. Conclusions and Future Work

We introduced in this paper a supervised learning based regression model *NowDeepN*, which used an ensemble of *deep artificial neural network* for predicting the values for meteorological products at a certain time moment based on their historical values. *NowDeepN* was intended to be a proof of concept for the feasibility of learning to approximate a function between past values of the radar products extracted from radar observations and their future values.

The *NowDeepN* model consisted of an ensemble of DNNs for radar products' values nowcasting. In this ensemble, a DNN model was used for learning to approximate the value of each radar product at a given time moment and a certain geographical location from the radar products' values from the neighborhood of that location at previous time moments.

Experiments were conducted on real radar data provided by the Romanian National Meteorological Administration and collected on the Central Transylvania region. The obtained results provided an empirical evidence that in both normal and severe weather conditions the values for a radar product at a given moment in a certain location are predictable from the values of the neighboring locations from previous time moments. This evidence is essential for further using convolutional neural network models for automatically extracting from radar data features which would be relevant for predicting the radar products' values at a certain time moment based on their historical values. A data cleaning step was introduced for correcting the erroneous input radar and its impact on increasing the predictive performance of the *NowDeepN* model was highlighted. In addition, the relevance of the features considered in the supervised learning task was empirically proven. More specifically, the experiments shown that the radar products' values from the neighboring area of a certain geographical location l at time $t-1$ are useful for predicting the radar products' values on location l at time t.

The experimental results highlighted that our *NowDeepN* model has a good performance particularly for high values of the reflectivity threshold, which indicate its ability to detect moderate and heavy precipitation and medium and large hail. While from a meteorological point of view, the performance of *NowDeepN* for predicting radar reflectivity values one time step ahead is satisfactory, in order to assess its performance compared to techniques currently employed in weather nowcasting, further development of the model for multiple time steps (e.g., 5 or 10 time steps, covering 30 or 60 min) is needed.

The experimental evaluation of *NowDeepN* will be further extended by enlarging the data set used for training the model. As future plans we aim to investigate convolutional neural network models [53] as well as supervised classifiers based on *relational association rule mining* [54,55] for detecting relationships between the meteorological products' values which may distinguish between normal and severe meteorological phenomena. In addition, we will analyze the possibility to extend the features used in the learning process, by combining radar data with other features (e.g., geographic and antropic features).

Author Contributions: Conceptualization, G.C. and A.M.; methodology, G.C. and A.M.; software, G.C. and A.M.; validation, G.C., A.M. and E.M.; formal analysis, G.C. and A.M.; investigation, G.C., A.M. and E.M.; resources, G.C., A.M. and E.M.; data curation, G.C., A.M. and E.M.; writing–original

draft preparation, G.C.; writing–review and editing, G.C., A.M. and E.M.; visualization, G.C., A.M. and E.M.; funding acquisition, G.C., A.M. and E.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research leading to these results has received funding from the NO Grants 2014–2021, under Project contract no. 26/2020.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from Romanian National Meteorological Administration and are available http://www.meteoromania.ro/ with the permission of Romanian National Meteorological Administration.

Acknowledgments: The research leading to these results has received funding from the NO Grants 2014–2021, under Project contract no. 26/2020. The authors acknowledge the assistance received from the National Meteorological Administration from Romania, for providing the meteorological data sets used in the experiments. The authors also thank the editor and the reviewers for their useful suggestions and comments that helped to improve the paper and the presentation.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Franch, G.; Nerini, D.; Pendesini, M.; Coviello, L.; Jurman, G.; Furlanello, C. Precipitation Nowcasting with Orographic Enhanced Stacked Generalization: Improving Deep Learning Predictions on Extreme Events. *Atmosphere* **2020**, *11*, 267. [CrossRef]
2. Chen, L.; Cao, Y.; Ma, L.; Zhang, J. A Deep Learning-Based Methodology for Precipitation Nowcasting With Radar. *Earth Space Sci.* **2020**, *7*, e2019EA000812. [CrossRef]
3. Swedish Meteorological and Hydrological Institute. Cooperation Is a Must for Adaptation to and Mitigation of Climate Change. 2018. Available online: https://www.smhi.se/en/news-archive/ (accessed on 10 July 2018).
4. WMO—World Meteorological Organisation. Weather Climate Water. 2018. Available online: https://www.wmo.int (accessed on 10 July 2018).
5. Kimura, R. Numerical weather prediction. *J. Wind. Eng. Ind. Aerodyn.* **2002**, *90*, 1403–1414. [CrossRef]
6. NOAA's Radar Operations Center. NEXRAD Technical Information, 2018. Available online: https://www.roc.noaa.gov/WSR88D/Engineering/NEXRADTechInfo.aspx (accessed on 10 July 2018).
7. Hering, A.M.; Morel, C.; Galli, G.; Sénési, S.; Ambrosetti, P.; Boscacci, M. Nowcasting thunderstorms in the Alpine region using a radar based adaptive thresholding scheme. In *Proceedings of ERAD*; Copernicus GmbH: Visby, Sweden, 2014; pp. 206–211.
8. James, P.; Reichert, B.; Heizenreder, D. NowCastMIX—optimized automatic warnings from continuously monitored nowcasting systems based on fuzzy-logic evaluations of storm attributes. In *European Conference on Severe Storms*; American Meteorological Society: Wiener Neustadt, Austria, 2015; pp. 1413–1433.
9. Merlet, N.; Cau, P.; Jauffret, C.; Maynard, K.; Sanchez, I.; Brousseau, P. AROME-NWC Overview, Results, Evolution and Perspectives. *Eur. Forecast.* **2017**, *22*, 40–43.
10. Dixon, M.; Wiener, G. TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A radar-based methodology. *J. Atmos. Ocean. Technol.* **1993**, *10*, 785–797. [CrossRef]
11. Johnson, J.T.; MacKeen, P.L.; Witt, A.; Mitchell, E.D.W.; Stumpf, G.J.; Eilts, M.D.; Thomas, K.W. The Storm Cell Identification and Tracking Algorithm: An Enhanced WSR-88D Algorithm. *Weather. Forecast.* **1998**, *13*, 263–276. [CrossRef]
12. Jung, S.H.; Lee, G. Radar-based cell tracking with fuzzy logic approach. *Meteorol. Appl.* **2015**, *22*, 716–730. [CrossRef]
13. Auger, L.; Dupont, O.; Hagelin, S.; Brousseau, P.; Brovelli, P. AROME—NWC: A new nowcasting tool based on an operational mesoscale forecasting system. *Q. J. R. Meteorol. Soc.* **2015**, *141*, 1603–1611. [CrossRef]
14. Haiden, T.; Kann, A.; Wittmann, C.; Pistotnik, G.; Bica, B.; Gruber, C. The Integrated Nowcasting through Comprehensive Analysis (INCA) System and Its Validation over the Eastern Alpine Region. *Weather. Forecast.* **2011**, *26*, 166–183. [CrossRef]
15. IBM. Deep Thunder, 2014. Available online: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepthunder/ (accessed on 15 September 2019).
16. Panasonic. Panasonic Global 4D Weather. 2016. Available online: https://www-media.panasonic.aero/2016/01/010616-PWS4DGlobal.pdf (accessed on 15 September 2019).
17. Climacell. The ClimaCell Engine, 2019. Available online: https://www.climacell.co/ (accessed on 15 September 2019).
18. TempoQuest. TempoQuest. 2015. Available online: http://tempoquest.com/ (accessed on 15 September 2019).
19. Mitchell, T.M. *Machine Learning*, 1st ed.; McGraw-Hill, Inc.: New York, NY, USA, 1997.
20. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
21. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
22. Gogas, P.; Papadimitriou, T.; Agrapetidou, A. Forecasting bank failures and stress testing: A machine learning approach. *Int. J. Forecast.* **2018**, *34*, 440–455. [CrossRef]

23. Tai, C.C.; Lin, H.W.; Chie, B.T.; Tung, C.Y. Predicting the failures of prediction markets: A procedure of decision making using classification models. *Int. J. Forecast.* **2019**, *35*, 297–312. [CrossRef]
24. Maymin, P.Z. Wage against the machine: A generalized deep-learning market test of dataset value. *Int. J. Forecast.* **2019**, *35*, 776–782. [CrossRef]
25. Nowlan, S.J.; Hinton, G.E. Simplifying Neural Networks by Soft Weight-Sharing. *Neural Comput.* **1992**, *4*, 473–493. [CrossRef]
26. Le, Q. Building high-level features using large scale unsupervised learning. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 8595–8598.
27. Deng, J.; Zhang, Z.; Marchi, E.; Schuller, B. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In Proceedings of the Humaine Association Conference on Affective Computing and Intelligent Interaction, Geneva, Switzerland, 2–5 September 2013; pp. 511–516.
28. Teletin, M.; Czibula, G.; Bocicor, M.I.; Albert, S.; Pandini, A. Deep Autoencoders for Additional Insight into Protein Dynamics. In *International Conference on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 79–89.
29. Teletin, M.; Czibula, G.; Codre, C. AutoSimP: An Approach for Predicting Proteins' Structural Similarities Using an Ensemble of Deep Autoencoders. In *International Conference on Knowledge Science, Engineering and Management*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 49–54.
30. Jolliffe, I.T.; Cadima, J. Principal component analysis: a review and recent developments. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2016**, *374*, 20150202. [CrossRef] [PubMed]
31. Han, L.; Sun, J.; Zhang, W.; Xiu, Y.; Feng, H.; Lin, Y. A machine learning nowcasting method based on real-time reanalysis data. *J. Geophys. Res. Atmos.* **2017**, *122*, 4038–4051. [CrossRef]
32. Beusch, L.; Clementi, L.; Foresti, L.; Hamann, U.; Hering, A.M.; Leonarduzzi, E.; Nerini, D.; Nisi, L.; Sassi, M.; Germann, U. Towards thunderstorm nowcasting by applying machine learning to a multi-sensor observation and NWP model database. In Proceedings of the European Conference on Severe Storms 2017, Pula, Croatia, 18–22 September 2017; Volume 136.
33. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.k.; Woo, W.c. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems—Volume 1*; MIT Press: Cambridge, MA, USA, 2015; pp. 802–810.
34. Kim, S.; Hong, S.; Joh, M.; Song, S. DeepRain: ConvLSTM Network for Precipitation Prediction using Multichannel Radar Data. *arXiv* **2017**, arXiv:1711.02316.
35. Heye, A.; Venkatesan, K.; Cain, J. Precipitation Nowcasting: Leveraging Deep Convolutional Recurrent Neural Networks. In *Proceedings of the Cray User Group (CUG)*; American Meteorological Society: Austin, Texas, 2017; pp. 1–8.
36. Shi, X.; Gao, Z.; Lausen, L.; Wang, H.; Yeung, D.; Wong, W.; Woo, W. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5622–5632.
37. Narejo, S.; Pasero, E. Meteonowcasting using Deep Learning Architecture. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 16–23. [CrossRef]
38. Sprenger, M.; Schemm, S.; Oechslin, R.; Jenkner, J. Nowcasting Foehn Wind Events Using the AdaBoost Machine Learning Algorithm. *Weather Forecast.* **2017**, *32*, 1079–1099. [CrossRef]
39. Ji, Y. Short-term Precipitation Prediction Based on a Neural Network. In Proceedings of the 3rd International Conference on Artificial Intelligence and Industrial Engineering, AIIE 2017, Shanghai, China, 26–27 September 2017; pp. 246–251.
40. Tran, Q.K.; Song, S.k. Computer Vision in Precipitation Nowcasting: Applying Image Quality Assessment Metrics for Training Deep Neural Networks. *Atmosphere* **2019**, *10*, 244. [CrossRef]
41. Han, L.; Sun, J.; Zhang, W. Convolutional Neural Network for Convective Storm Nowcasting Using 3D Doppler Weather Radar Data. *IEEE Trans. Geosci. Remote. Sens.* **2020**, *58*, 1487–1495. [CrossRef]
42. Yan, Q.; Ji, F.; Miao, K.C.; Wu, Q.; Xia, Y.; Li, T. Convolutional Residual-Attention: A Deep Learning Approach for Precipitation Nowcasting. *Adv. Meteorol.* **2020**, *2020*, 1–12. [CrossRef]
43. Dietterich, T.G. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Mach. Learn.* **2000**, *40*, 139–157. [CrossRef]
44. Mass, C. Nowcasting: The Promise of New Technologies of Communication, Modeling, and Observation. *Bull. Am. Meteorol. Soc.* **2012**, *93*, 797–809. [CrossRef]
45. Czibula, G.; Mihai, A.; Mihuleţ, E.; Teodorovici, D. Using self-organizing maps for unsupervised analysis of radar data for nowcasting purposes. *Procedia Comput. Sci.* **2019**, *159*, 48–57. [CrossRef]
46. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). *arXiv* **2018**, arXiv:1803.08375.
47. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
48. Hyndman, R.J.; Koehler, A.B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688. [CrossRef]
49. Brown, L.; Cat, T.; DasGupta, A. Interval Estimation for a proportion. *Stat. Sci.* **2001**, *16*, 101–133.
50. Keras. The Python Deep Learning Library. 2018. Available online: https://keras.io/ (accessed on 15 December 2020).
51. Mihai, A. NowDeepN Software. 2020. Available online: https://github.com/mihaiandrei1294/NowDeepN (accessed on 15 December 2020).
52. Scikit-learn. Machine Learning in Python. 2018. Available online: http://scikit-learn.org/stable/ (accessed on 15 December 2020).
53. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights Into Imaging* **2018**, *9*, 611–629. [CrossRef] [PubMed]

54. Miholca, D.L.; Czibula, G.; Czibula, I.G. A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. *Inf. Sci.* **2018**, *441*, 152–170. [CrossRef]
55. Crivei, L.M. Incremental relational association rule mining of educational data sets. *Stud. Univ. Babes-Bolyai Ser. Inform.* **2018**, *63*, 102–117. [CrossRef]

applied sciences

MDPI

Article

Comparison of Instance Selection and Construction Methods with Various Classifiers

Marcin Blachnik [1,*] and Mirosław Kordos [2]

[1] Faculty of Materials Engineering, Department of Industrial Informatics, Silesian University of Technology, Akademicka 2A, 44-100 Gliwice, Poland; marcin.blachnik@polsl.pl

[2] Department of Computer Science, University of Bielsko-Biała, Willowa 2, 43-309 Bielsko-Biała, Poland; mkordos@ath.bielsko.pl

* Correspondence: marcin.blachnik@polsl.pl

Received: 7 May 2020; Accepted: 1 June 2020; Published: 5 June 2020

Abstract: Instance selection and construction methods were originally designed to improve the performance of the k-nearest neighbors classifier by increasing its speed and improving the classification accuracy. These goals were achieved by eliminating redundant and noisy samples, thus reducing the size of the training set. In this paper, the performance of instance selection methods is investigated in terms of classification accuracy and reduction of training set size. The classification accuracy of the following classifiers is evaluated: decision trees, random forest, Naive Bayes, linear model, support vector machine and k-nearest neighbors. The obtained results indicate that for the most of the classifiers compressing the training set affects prediction performance and only a small group of instance selection methods can be recommended as a general purpose preprocessing step. These are learning vector quantization based algorithms, along with the *Drop2* and *Drop3*. Other methods are less efficient or provide low compression ratio.

Keywords: machine learning; classification; preprocessing; instance selection

1. Introduction

Classification is one of the basic machine learning problems, with many practical applications in industry and other fields. The typical process of constructing a classifier consists of data collection, data preprocessing, training and optimizing the prediction models and finally applying the best of the evaluated models. The described scheme is obvious, however we face two types of problems. The first one is that recently we more often start to construct classifiers with limited resources and the second one is that we want to interpret and understand the data and the constructed model easily.

The first group of restrictions are mostly related to time and memory constraints, where machine learning algorithms are often trained on mobile devices or micro computers like Rasberry Pi and other similar devices. There are basically three approaches to overcome these restrictions:

- using specially tailored algorithms which are designed to face those challenge
- limiting the size of the training data
- exporting the training process from the device to the cloud or other high performance environment

In the paper we focus on the second approach where instead of redesigning the classification algorithm or sending the data to the cloud we analyze how the data filters or in other words how the training set reduction methods influence classification performance with the data processing pipeline depicted in Figure 1.

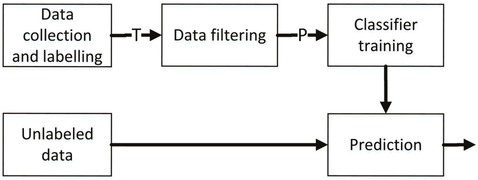


Figure 1. The pipeline of prediction model construction with data filtering.

The data filter has two goals: first it can improve the classifier performance by eliminating noisy samples from the training data thus allowing to achieve higher classification accuracy, and the second goal is training set compression. Training set size reduction allows to speed up classifier construction process but also it speeds up decision making when the classifier is already trained [1]. The speed up in classifier training is rather obvious when the size of the input data is smaller but the speed up in the prediction phase results from smaller number of support vectors of the support vector machine, shallow trees (earlier stopping of tree construction) and lower number of reference vectors in *k*NN. Moreover, it speeds up model selection and optimization. Here, the gain should be multiplied by the number of evaluated classifiers and their hyper-parameters, because the filtering stage is applied once and then the classifier selection and optimization is carried out.

Training set compression can be also used for solving model interpretability where the so called prototype-based rules [2,3] can be applied. These rules are also based on limiting the size of the data. And in this case, even if the original dataset is small, further limiting dataset size is still beneficial.

The main purpose of the paper and the research objective is the analysis of both aspects of data filtering that is the influence on prediction accuracy of various classifiers and the influence on training set size reduction.

In the study we evaluate a set of 20 most popular instance selection and construction methods used as data filters and 7 popular classifiers on 40 datasets in terms of classification performance and training set compression. The evaluated data filters are: condensed nearest neighbor (*CNN*), edited nearest neighbor (*ENN*), repeated-edited nearest neighbor (*RENN*), *All-kNN*, instance based learning version 2 (*IB2*), relative neighbor graph editing (*RNGE*), *Drop* family, iterative case filtering (*ICF*), modified selective subset selection (*MSS*), hit-miss network editing (*HMN-E*), hit-miss network interactive editing (*HMN-EI*), class conditional instance selection (*CCIS*) as well as learning vector quantization version 1,2.1 (respectively *LVQ1*, *LVQ2.1*), generalized learning vector quantization (*GLVQ*), *k-Means*. These data filters are selected for the evaluation, because they are the most frequently used instance selection methods. The datasets after filtering are used to train classifiers such as k-nearest neighbor (*k*NN), support vector machine (SVM), decision tree based methods, linear model and simple Naive Bayes. The hyperparameters of each of these classifiers are optimized with the grid search approach in order to achieve the highest possible prediction accuracy on the compressed data. Moreover, the obtained results are compared to the results obtained with simple stratified random sampling, which defines an acceptance threshold below which particular methods are not beneficial for given classifiers.

The article is structured as follow: in the next section an overview of instance selection methods is provided, with a literature overview, and also the research gap is presented, then in Section 3 we describe the experimental setup, and in Section 4 the results are presented. Finally, Section 5 summarizes the paper with general conclusions.

2. Research Background

2.1. Problem Statement

From the statistical point of view, reduction of the training set size will not affect prediction accuracy of the final classifier when the conditional probability $P(c|\mathbf{x})$ of predicting class c for given vector **x** remains unchanged when estimated from the original training set **T** and from the set of prototypes **P** obtained from the data filtering process.

In the literature one of popular multidimensional probability estimation methods is based on the nearest neighbor approach [4]. Similarly, for *k*NN many data filtering methods were developed in order to select a suitable subset of the training set. These methods are called instance selection methods and instance construction or prototype generation methods, and mostly they were designed to overcome weaknesses of the *k*NN classifier. In instance selection, usually the performance of *k*NN or even 1-NN classifier is used to identify those training samples which are important for the classification. These are mostly border samples close to the decision boundary. Instances which represent larger groups of instances from the same class and noise samples are usually filtered out because they reduce the performance of *k*NN. On the other hand, instance construction methods tend to find optimal position of the stored samples by 1-NN, so they do not need to represent samples from the original training set, these are usually new samples. An example effect of applying instance selection methods to the training set is presented on Figure 2.

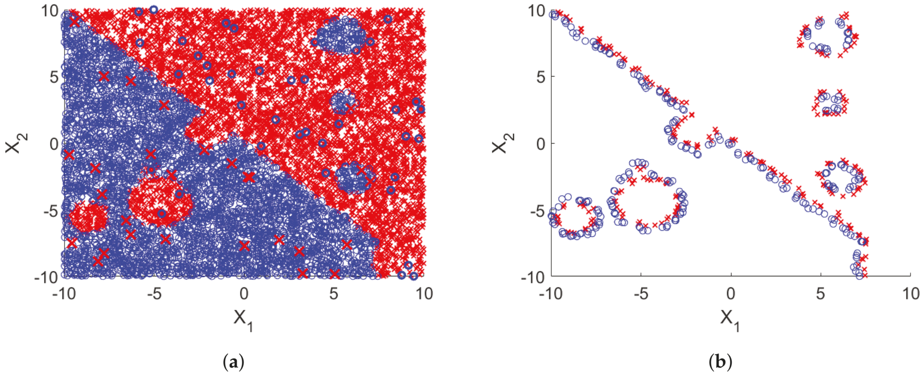


Figure 2. Example effect of artificial 2D training set compression using *Drop3* instance selection methods. (**a**) Original training set with additional noise (**b**) Training set after compression.

The idea of applying instance selection and prototype generation methods as data filters is not new and it is often considered a standard preprocessing step. In particular in Reference [5] some of the instance selection methods evaluated in our study were considered as the most effective preprocessing methods.

For example we have applied instance selection in the optimization of metallurgical processes for data size limitation and rule extraction [6,7], but instance selection methods were also applied in haptic modeling [8] as well as for automatic machine learning [9]. However, we cannot find in the literature any comprehensive study analyzing the influence of the instance selection methods on various classifiers. Most of the authors when presenting new algorithms indicate only the performance of 1-NN, *k*NN with fixed *k* parameter, and sometimes other classifiers but usually also with fixed hyperparameters. Such a comparison can be considered unfair, because the training set is being changed during the data filtering so different parameters are required by the classifiers. Unfortunately, such a comparison requires much larger computational time especially when using grid search with internal cross-validation procedure, so the process is usually simplified and only classifiers with fixed parameters are used. Only in References [10,11] some broader comparison is available but the experiments were conducted on only few (6 and 8) datasets. To fill that gap we provide a detailed analysis of the influence of the data based on instance selection and construction methods applied to 40 datasets.

2.2. Instance Selection and Construction Methods Overview

One of the most important properties of data filtering methods is the relation between instances in the original training set **T** and the in the selected prototype set **P**. If $\mathbf{P} \subset \mathbf{T}$ then the methods are called instance selection algorithms, because the prototypes **P** are selected directly from the training set **T**. This property does not hold for prototype construction also called prototype generation methods. In this case the elements of **P** are new vectors which can constitute completely new instances, which have never appeared in **T**.

This property is important considering the comprehensibility of the selected samples. In the case of instance selection methods the instances can be mapped into real objects, while in the case of instance construction methods the direct mapping is not possible. This is especially important when working with prototype-based rules [2,12], or other interpretable models.

In the literature perhaps the best overview of instance selection methods can be found in Reference [13] where the authors provide a taxonomy of over 70 instance selection methods, and empirically compare half of them in terms of compression and prediction performance of 1-NN classifier. The same group of authors of Reference [14] perform a similar analysis for prototype generation methods where 32 methods are discussed and 18 of them are empirically compared in application only for 1-NN classifier.

The taxonomy of data filtering methods can be presented in the following aspects:

- **search direction**
  - **incremental**, when given method starts from an empty set **P** and iteratively adds new samples, such as in *CNN* [15] or *IB2* [16]
  - **decremental**, when a given method starts from $\mathbf{P} = \mathbf{T}$ and then samples from **P** are iteratively removed such as in *HMN-E* and *HMN-EI* [17], *MSS* [18], *Drop1*, *Drop2*, *Drop3* and *Drop5* [19], *RENN* [20].
  - **batch**, when the instances are removed at once after analysis of the input training set. An examples of such methods are *ENN* [21], *All-kNN* [20] *RNGE* [22], *ICF* [23], *CCIS* [24].
  - **fixed**, when a fixed number of prototypes is given as the hyperprarametr of the method. This group includes *LVQ* (*LVQ1*, *LVQ2.1*, *GLVQ*) family [25,26], *k-Means* and random sampling.
- **type of selection**
  - **condensation**, when the algorithm tries to remove samples, which are located far from the decision boundary, as in the case of *IB2*, *CNN*, *Drop1* and *Drop2*, *MSS*.
  - **edition**, when the algorithm is designed for noise removal, such as *ENN*, *RENN*, *All-kNN*, *HMN-E*.
  - **hybrid**, when the algorithm performs both steps—condensation and edition. Usually these methods starts from noise removal, and then perform condensation. This group includes: *Drop3*, *Drop5*, *ICF*, *HMN-EI*.
- **the evaluation method**
  - **filters**, where the method uses internal heuristic independent to the final classifier.
  - **wrappers**, when external dedicated classifier is used to identify important samples.

The decision of assigning an algorithm to the right evaluation method depends on the final prediction model applied after data filtering. If the instance selection or construction method is followed by 1-NN or *k*NN classifier they can be considered as wrappers, because internally all of them use a kind of nearest neighbor based approach to decide whether an instance should be selected or rejected. On the other hand they can be also considered as filters, when the data filter takes as input entire training set and returns selected subset which is then used to train any classifier, not only the *k*NN. There are implementations which works as wrappers, so they allow to use all kind of classifiers such as in Reference [27], where instead of *k*NN any other classifier can be used, in particular the MLP neural network was used. The drawback of the wrappers is the increase of computational complexity. Here in this article we only consider the standard instance selection methods without any generalization.

3. Experimental Design

There are several factors which determine the applicability of given algorithms as a general purpose training set filter. Among the most important are compression level and prediction accuracy of the final classifier. The compression is defined as:

$$comp = 1 - \frac{\|\mathbf{P}\|}{\|\mathbf{T}\|}, \tag{1}$$

so that higher value of compression indicates that more samples were rejected and the resultant set **P** is smaller and lower values (close to 0) indicates that the output training set is larger. The second property is the prediction accuracy of the classifier trained on **P**. This value is subjective to the applied classifier, so that for one classifier given set **P** may result in high accuracy, while for the other the accuracy can be worse. Here a simple accuracy measure was evaluated:

$$acc = \frac{\text{\#correctly classified samples}}{\text{\#all evaluated samples}}. \tag{2}$$

In order to determine the applicability of instance selection and construction methods as universal training set filters we designed experiments which mimic typical use cases of training set filtering. The scheme of the data processing pipeline is presented in Figure 3

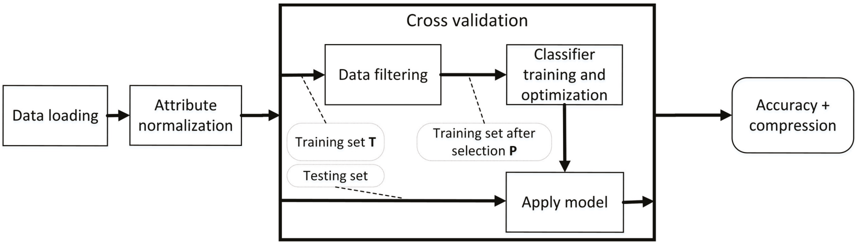


Figure 3. The pipeline of data processing used in the experiments.

It starts with data loading and attribute normalization, then the 10 fold cross-validation procedure is executed which wraps the data filtering stage (our instance selection or construction method) followed by classifier training and hyperparameter optimization procedure. Finally, the trained classifier is applied to the test set. During the process execution prediction accuracy and compression were recorded.

In the experiments the most commonly used classifiers were evaluated. These are: the basic classifiers for which the evaluated data filters were designed such as 1-NN and *k*NN; simple classifiers like Naive Bayes or linear model (GLM); followed by kernel methods such as SVM with Gaussian kernel and finally the decision tree based methods including C4.5 and Random Forest. Many of these methods require careful parameter selection such as the value of k in *k*NN or C and γ in SVM or the number of trees in Random Forest. All of the evaluated parameters are presented in Table 1. It is important to note that each of the applied data filters was independently evaluated for each classifier, because a particular filter may be beneficial for one classifier and unfavorable for another.

Table 1. Parameter settings of the evaluated classifiers.

Classifier	Parameter	Values	Implementation
1-NN	-	-	RapidMiner
*k*NN	k	1:2:40	RapidMiner
Naive Bayes	-	-	RapidMiner
GLM	-	-	H_2O
C4.5	-	-	Weka
Random Forest	# trees	$\{20, 40, 60, 80, 100\}$	Weka
SVM	C γ	$1E\{-1, 0, 1, 2\}$ $\{1, 3, 5, 7, 9\}$	LibSVM

The entire group of instance selection and construction methods is very broad. As indicated in Section 2.2 some authors distinguish over 70 instance selection methods and over 32 prototype construction methods [13,14]. From these groups we selected the most popular ones which can be found in many research papers as the reference methods [10,11,28–30]. These are *CNN*, *ENN*, *RENN*, *All-kNN*, *IB2*, *RNGE*, *Drop1*, *Drop2*, *Drop3*, *Drop5*, *ICF*, *MSS*, *HMN-E*, *HMN-EI*, *CCIS*, from the group of instance selection methods, and from the group of prototype generation methods we selected 3 algorithm from the family of *LVQ* algorithms, these are *LVQ1*, *LVQ2.1*, as well as the *GLVQ* algorithm. In the experiments we also evaluated *k-Means* clustering algorithm which is most often used to reduce the size of the training set [31]. The *k-Means* algorithm was independently applied to each class label, and then the obtained cluster centers were used as prototypes with appropriate class labels [32]. All of the prototype generation methods belong to the group of fixed methods, so they require to determine the compression manually. For that purpose the experiments were carried out for two different initial sets of prototypes: randomly selected 10% of the training samples used for initialization which corresponds to 90% compression and also 30% of the training samples which corresponds to 70% compression. The 90% compression is the lower bound of the compression obtained by most of instance selection methods.

All evaluated methods were also compared with the random stratified sampling (*Rnd*), which is the simplest solution that can be used as a data filter. Similarly as with prototype construction methods, the experiments with *Rnd* were conducted for compression 70% and 90% that corresponds to *Rnd(0.3)* and *Rnd(0.1)* (the numbers represent percentage of the samples that remain). The accuracy obtained for *Rnd* constitute the lower bound which allows to distinguish beneficial data filters from the weak ones that are worse than simple random sampling.

The experiments were carried out on 40 datasets obtained from the Keel repository [33]. A list of the datasets is presented in Table 2. All the calculations were conducted using RapidMiner software with Information Selection extension developed by the authors [34]. The extension is available at the RapidMiner Marketplace and the most recent version is also available at GitHub (https://github.com/mblachnik/infoSel). Some of the evaluated algorithms like *HMN-EI* and *CCIS* were taken from the Keel framework [35] and integrated with the Information Selection extension.

Table 2. Datasets used in the experiments. The s/a/c denotes the number of samples, attributes and classes.

Id.	Name	s / a / c	Id.	Name	s / a / c
1	appendicitis	106 / 7 / 2	21	page-blocks	5472 / 10 / 5
2	balance	625 / 4 / 3	22	phoneme	5404 / 5 / 2
3	banana	5300 / 2 / 2	23	pima	768 / 8 / 2
4	bands	365 / 19 / 2	24	ring	7400 / 20 / 2
5	bupa	345 / 6 / 2	25	satimage	6435 / 36 / 6
6	cleveland	297 / 13 / 5	26	segment	2310 / 19 / 7
7	glass	214 / 9 / 6	27	sonar	208 / 60 / 2
8	haberman	306 / 3 / 2	28	spambase	4597 / 57 / 2
9	hayes-roth	160 / 4 / 3	29	spectfheart	267 / 44 / 2
10	heart	270 / 13 / 2	30	tae	151 / 5 / 3
11	hepatitis	80 / 19 / 2	31	texture	5500 / 40 / 11
12	ionosphere	351 / 33 / 2	32	thyroid	7200 / 21 / 3
13	iris	150 / 4 / 3	33	titanic	2201 / 3 / 2
14	led7digit	500 / 7 / 10	34	twonorm	7400 / 20 / 2
15	mammographic	830 / 5 / 2	35	vehicle	846 / 18 / 4
16	marketing	6876 / 13 / 9	36	vowel	990 / 13 / 11
17	monk-2	432 / 6 / 2	37	wdbc	569 / 30 / 2
18	movement_libras	360 / 90 / 15	38	wine	178 / 13 / 3
19	newthyroid	215 / 5 / 3	39	wisconsin	683 / 9 / 2
20	optdigits	5620 / 64 / 10	40	yeast	1484 / 8 / 10

4. Results and Analysis

Since simple averaging has limited interpretability, we used both average performance and average rank to asses the quality of the evaluated methods. In order to make ranking for each dataset and each classifier the results obtained for particular data filters were ranked from the best to the worst in terms of classification accuracy and compression. The highest rank (equal to 26, which is the number of evaluated algorithms) was given to the best filter method for particular dataset and the lowest rank (1) was assigned to the worst method (rank with ties). Then the ranks over datasets were averaged to indicate the final performance. Such a comparison does not reflect how much one method differs from the other in terms of given quality measure, but ranking unlike averaging performances is insensitive to the data distribution where measures like accuracy can range from 40% on one dataset up to 99% on another. On the other hand ranking do not provide information on how much the methods differ so these both quality measures complement each other and should be considered together, where the ranking gives an answer which method was more often better, and then, the mean performance indicates how much given method was better from the competitor. Moreover, the threshold obtained by the random sampling should be applied simultaneously to both values and when any of them is below the threshold given method should be rejected as useless.

The obtained results including both average ranks as well as average performances are presented in Table 3. Moreover, the Wilcoxon signed-rank statistical test [36] was used to check whether the results obtained by the classifier without any data filter significantly differ from the results obtained when given data filter was used to cleanup the dataset. The calculations were conducted with significance level $\alpha = 0.1$. The data filters which did not lead to a significantly decrease of the prediction accuracy were marked with =. In few cases the data filter allowed to increase the accuracy of a classifier, and if the increase was statistically significant we marked the results with + sign. In this case the significance was measured using Wilcoxon tailed sign-rank test.

Table 3. Average rank of accuracy and compression obtained for data filtering methods for given classifiers. The symbols +,= indicate the results of Wilcoxon sign-rank test. = represents not significant difference in comparison to *No filter*, + represents significant positive difference in comparison to *No filter*.

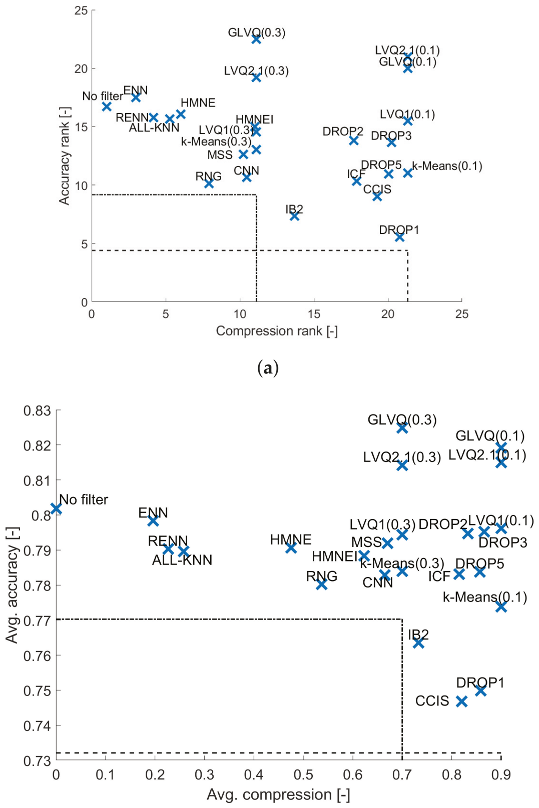
Compression Method	**Compression**		**1-NN**		***k*NN**		**Naive Bayes**		**GLM**		**C4.5**		**Random Forest**		**SVM**	
	Rnk [-]	**Cmp** [%]	**Rnk** [-]	**Acc** [%]	**Rnk** [-]	**Acc** [%]	**Rnk** [-]	**Acc** [%]	**Rnk** [-]	**Acc** [%]	**Rnk** [-]	**Acc** [%]	**Rnk** [-]	**Acc** [%]	**Rnk** [-]	**Acc** [%]
ENN	2.97	19.66	17.50	79.74=	17.37	81.44	19.46	74.43+	18.79	78.72	22.18	79.61=	20.87	83.46	18.08	83.70
RENN	4.18	22.74	15.76	78.75=	14.36	80.43	18.41	74.51+	15.73	77.57	20.37	78.05	17.18	81.46	13.13	82.24
ALL-KNN	5.28	26.02	15.64	78.66=	12.85	79.55	17.62	74.46=	13.78	77.29	20.45	78.17=	15.45	81.59	13.41	82.28
HMNE	6.03	46.45	16.05	79.22=	16.74	81.42	17.04	72.71=	18.24	78.52	20.42	78.22	17.63	82.67	17.96	83.90
RNG	7.91	53.25	10.14	78.18	15.13	82.38	12.78	68.46	17.23	77.09	14.23	74.34	19.22	83.72	18.42	84.53
HMNEI	11.00	60.86	15.01	79.00=	12.58	80.82	14.42	70.48=	15.41	76.61	15.53	74.30	13.86	79.57	13.28	82.47
CNN	10.46	66.38	10.65	78.42	14.74	82.03	10.74	67.53	13.65	74.28	12.51	72.13	16.03	81.28	17.77	82.96
MSS	10.23	66.88	12.63	79.35	16.40	82.70	10.24	67.68	13.24	74.37	11.58	71.45	15.63	80.87	15.60	82.75
IB2	13.69	73.11	7.33	76.49	10.13	80.69	8.14	66.55	12.05	73.58	8.63	70.13	11.58	79.90	13.72	82.20
ICF	17.87	80.84	10.33	78.22	9.72	80.40	11.31	69.22	11.08	73.41	11.85	71.72	12.42	79.18	11.42	81.11
CCIS	19.28	80.91	9.03	74.78	7.31	76.48	12.18	67.85	10.12	73.25	11.14	69.96	8.55	76.81	7.77	78.74
DROP2	17.69	82.94	13.82	79.44=	11.96	80.92	11.82	68.07	11.74	73.47	13.18	71.94	13.41	79.32	12.23	81.32
DROP5	20.06	85.28	10.94	78.37	10.68	80.38	10.62	69.14	11.29	73.42	10.90	71.07	10.90	78.84	10.99	81.12
DROP1	20.78	85.47	5.55	74.83	5.29	77.38	7.77	64.97	7.59	71.22	7.88	68.18	7.91	76.47	6.35	77.88
DROP3	20.23	85.99	13.64	79.30=	10.87	80.60	11.85	69.15	12.35	75.10=	12.06	72.15	12.58	79.71	10.27	81.28
GLVQ(0.1)	21.33	90.00	19.97	81.70+	16.27	82.15	13.47	70.56=	10.82	74.51	5.21	62.46	5.47	72.96	9.41	80.58
GLVQ(0.3)	11.13	70.00	22.47	82.49+	21.85	83.53=	18.42	73.77=	15.45	76.60	12.87	71.93	12.38	79.22	16.50	83.68
LVQ1(0.1)	21.33	90.00	15.47	79.19=	10.90	79.78	13.47	70.69=	9.94	74.62	9.44	70.04	9.51	78.27	9.99	81.10
LVQ1(0.3)	11.13	70.00	14.53	79.42=	15.85	81.62	14.38	72.01	17.03	77.60	15.09	75.54	17.00	82.04	18.09	83.68
LVQ2.1(0.1)	21.33	90.00	20.96	81.33+	17.69	81.79	13.71	69.29=	10.56	74.17	6.45	64.71	6.27	73.20	9.41	79.26
LVQ2.1(0.3)	11.13	70.00	19.21	81.45+	19.53	82.97=	15.35	72.82=	15.82	76.95	16.06	74.13	15.77	80.72	16.90	83.40
k-Means(0.1)	21.33	90.00	11.04	77.09	9.37	78.83	11.96	69.81=	10.10	74.76	9.08	70.60	9.63	78.06	10.73	81.12
k-Means(0.3)	11.13	70.00	13.04	78.41	16.04	81.24	14.38	71.05=	15.19	76.97	15.04	75.15	16.05	81.21	18.29	83.24
Rnd(0.1)	21.33	90.00	4.40	72.49	3.41	75.07	11.74	69.94	7.97	73.50	9.54	70.95	7.71	77.25	4.71	77.89
Rnd(0.3)	11.13	70.00	9.17	76.77	11.45	79.67	12.65	71.14	14.99	77.11	16.90	76.30	15.64	81.43	13.95	82.31
No filter	1.01	0.00	16.72	80.33	22.53	83.70	17.05	72.16	20.82	79.00	22.42	79.86	22.36	84.79	22.63	85.65

To increase readability, the results which represent ranks are also presented graphically independently for each classifier. In the figures the doted lines represent the performances obtained by random sampling, so that if any filter method appears within the space defined by the doted lines it is dominated by simple random sampling (*Rnd*).

Below in the following subsection the term "reference method" is used to describe the algorithm without data filter, this is the classifier which was directly applied to the training data.

4.1. 1-NN

The results obtained for 1-NN are visualized in Figure 4. They indicate that the *GLVQ* and *LVQ2.1* significantly outperform other methods and especially the reference solution without any data filter. From the group of instance selection methods the best ones are noise filters *ENN*, *HMN-E* and from the group of condensation methods—*Drop2* and *Drop3* are dominating. It is also noticeable that all of the data filters appear above the base rates defined by the random sampling.



(**a**)

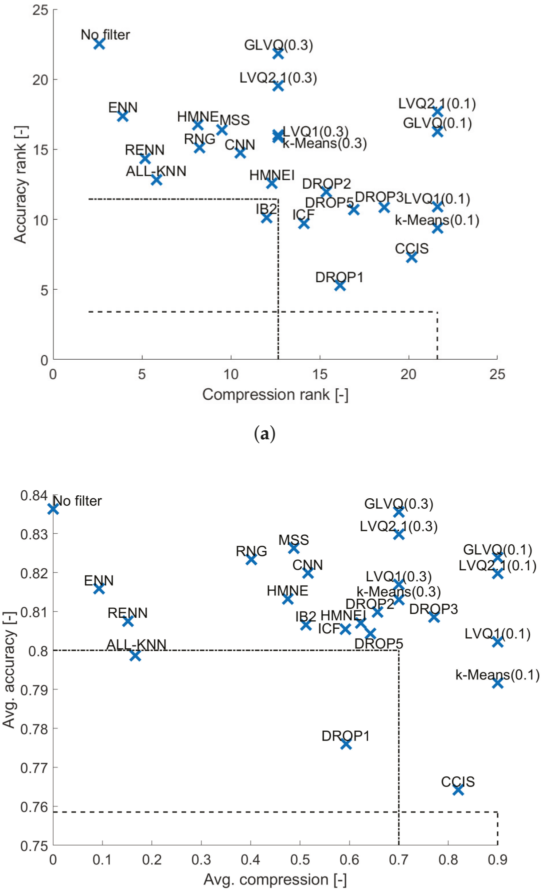
(**b**)

Figure 4. Results obtained for 1-NN classifier. (**a**) Average performance ranks. (**b**) Average performance.

*4.2. k*NN

In the case of *k*NN similarly the best results are obtained for *GLVQ* and *LVQ2.1* (see Figure 5), but they do not differ significantly from the results obtained by *k*NN with optimally tuned *k* parameter. All other filters appears to decrease classification accuracy. Also noticeable is the fact that now

IB2 appears to be dominated by the random sampling, as well as *All-kNN* and *Drop1* in terms of average accuracy.



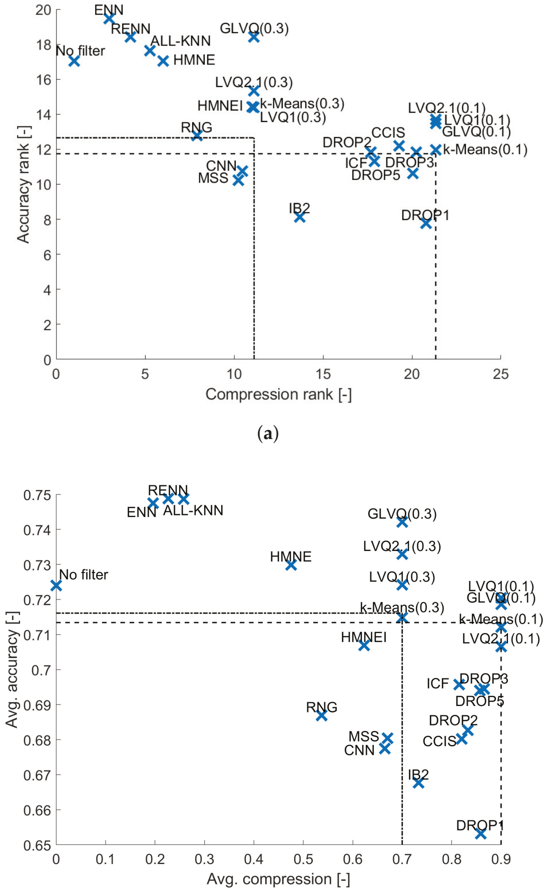
(**a**)

(**b**)

Figure 5. Results obtained for *k*NN classifier. (**a**) Average performance ranks (**b**) Average performance.

4.3. Naive Bayes

The results obtained for Naive Bayes are shown in Figure 6. There is no significant difference between the accuracy obtained for the two random sampling methods (one with compression 90% and the second with compression 70%). The difference between these two in terms of ranks is less than 1. For Naive Bayes the highest accuracy is obtained by the *ENN* and *RENN*, for which the comparison to the reference method is statistically significantly different. Also *All-kNN* is very high, but the Wilcoxon test does not indicate significant statistical difference. That is reasonable because noise filters remove the noise samples which affects the probability distributions estimated by the Naive Bayes classifier. Here the *LVQ* family, especially the *GLVQ*(0.3) algorithm, displays similar performance to noise filters, but with the compression reaching 70%, unfortunately the difference to the reference method is not significant. Also other prototype construction methods like other *LVQ* algorithms as well as *k-Means* clustering method do not show significant differences. From the group of evaluated methods almost all instance selection algorithms are dominated by the random sampling so all these methods can be considered as unhelpful.



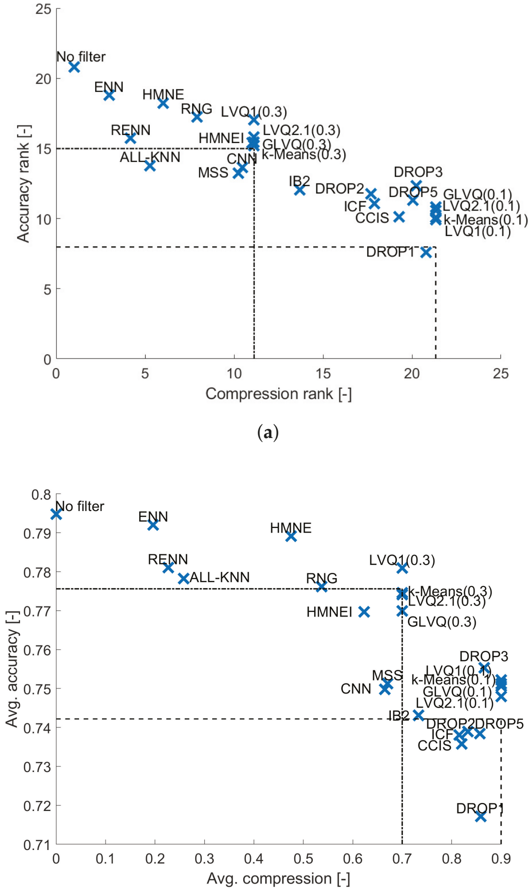
(**a**)

(**b**)

Figure 6. Results obtained for Naive Bayes classifier. (**a**) Average performance ranks. (**b**) Average performance.

4.4. GLM

The linear model without instance selection provided the highest accuracy, as shown in Figure 7 and by applying any data filter we may expect a drop in accuracy. The highest accuracy using filters is obtained for *ENN* and *HMN-E* and for larger compression methods with *LVQ1*, *GLVQ* and *Drop3*, but all these results are statistically significantly different. It is important to mention that the linear model can be efficiently implemented, so the data filters are not necessarily required, because they extend the computation time. For GLM nine models are dominated by random sampling, these are *Drop1*, *Drop2*, *Drop5*, *ICF*, *CCIS*, *CNN*, *MSS*, *HMN-EI* and *All-kNN* and many are close to the border like *GLVQ*, *LVQ2.1* or *k-Means*, so in general it is not recommended to perform any data filtering for the GLM model.



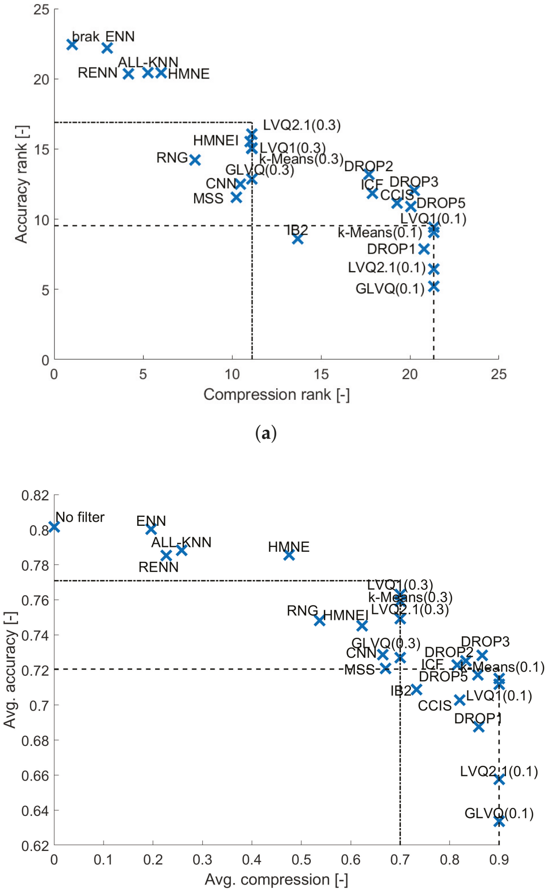
(**a**)

(**b**)

Figure 7. Results obtained for GLM classifier. (**a**) Average performance ranks. (**b**) Average performance.

4.5. C4.5 Decision Tree

In the case of C4.5 decision tree (see Figure 8) it could be expected that any dataset reduction may result in the drop of accuracy. This is due to the quality of estimated statistics which are determined when selecting the split nodes. As a result the majority of data filters are dominated by random sampling. Only noise filters allows to achieve the accuracy comparable to the one obtained by the reference method, moreover the results for *ENN* and *All-kNN* are not statistically significantly different. This is due to the fact that noise filters have very low compression, but also regularizing the decision boundary by eliminating the noise samples can have positive influence on the estimated measures of decision tree nodes quality. For the condensation methods only *Drop3*, *Drop2* and *ICF* achieve results not dominated by random sampling.



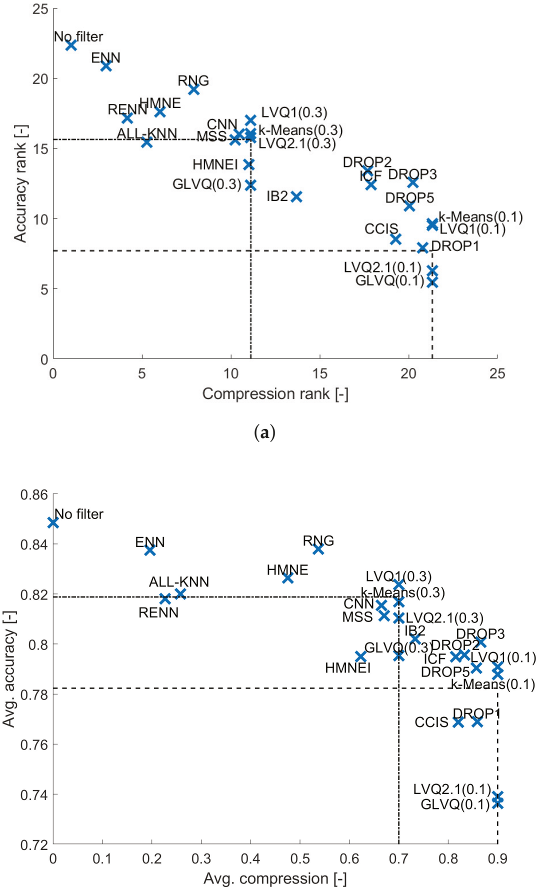
(**a**)

(**b**)

Figure 8. Results obtained for C4.5 classifier. (**a**) Average performance ranks. (**b**) Average performance.

4.6. Random Forest

Random Forest is a classifier which is also based on the decision tree but thanks to the properties of collective decision making it can overcome some of its weaknesses. As shown in Figure 9 only *ENN* achieves comparable accuracy to the one obtained with entire training set. However, almost all data filtering methods especially instance selection methods are better than random sampling (except *HMN-EI* and *All-kNN* which lie on the border), and almost all prototype construction methods (except *LVQ1*) lie on the bounds defined by random sampling. Note that here all methods are statistically significantly different from the reference method, that is worse than the reference solution.



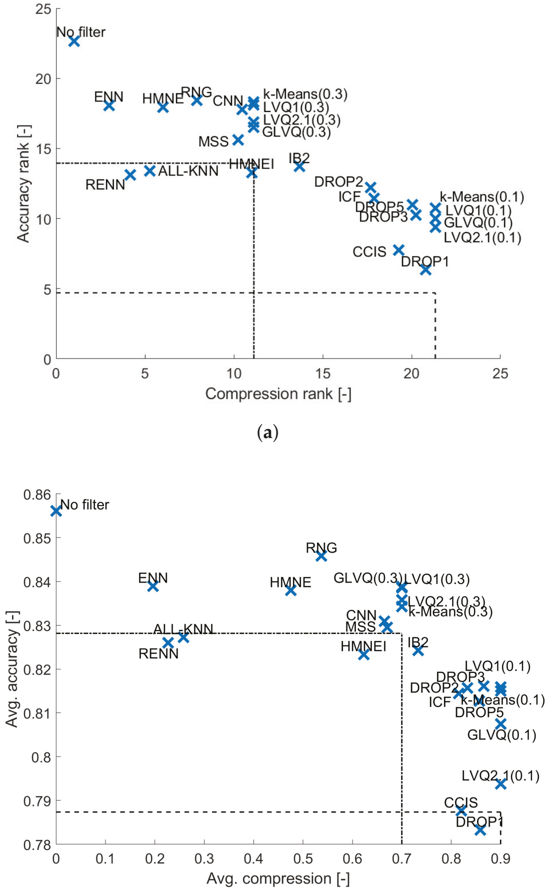
(**a**)

(**b**)

Figure 9. Results obtained for Random Forest classifier. (**a**) Average performance ranks. (**b**) Average performance.

4.7. SVM

The final of the evaluated classifiers is SVM which is one of the most robust classifiers (similarly to Random Forest). The results presented in Figure 10 indicate that all the examined instance selection methods lead to decrease in prediction accuracy. Moreover, for the compression level of up to 70% the top data filters are *ENN*, *HMN-E*, *RNGE*, *CNN*, *k-Means* and *LVQ1*, which on average share similar accuracy rank. Further increase in compression leads to significant drop in accuracy rank, so the best methods with compression equal 90% like *k-Means* and *LVQ1* have accuracy rank almost 8 points lower. For SVM only *RENN*, *All-kNN* and *HMN-EI* (which all belong to the noise filters) are outperformed by random sampling. The reason fo that are the figh tolerance on noise in the data that can be controlled by *C* parameter in SVM.



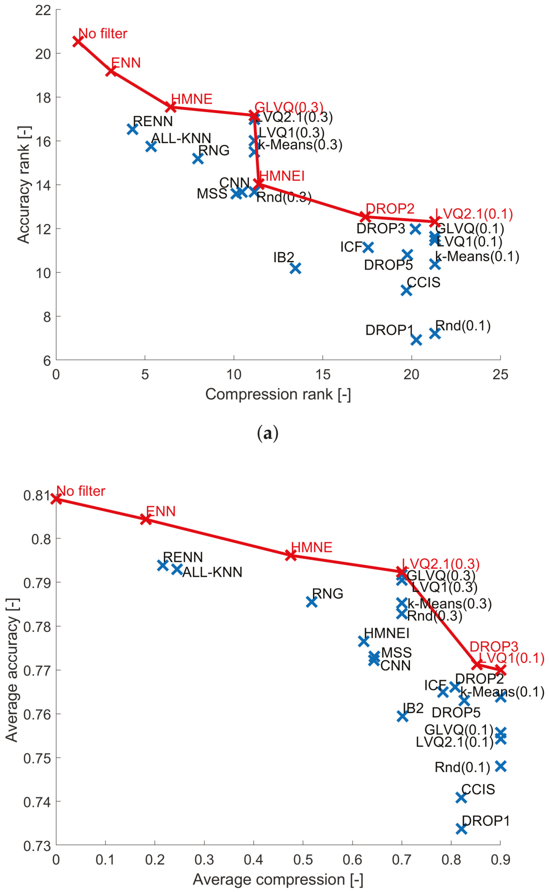
(**a**)

(**b**)

Figure 10. Results obtained for SVM classifier. (**a**) Average performance ranks. (**b**) Average performance.

5. Conclusions

In the article we investigated the performance of the popular classical instance selection and prototype generation methods in terms of the obtained compression of the data set and their influence on the performance of various classifiers. To summarize the obtained results we averaged them for each data filtering method over all classifiers. This allowed us to compare all the evaluated data filters. The results are presented in Figure 11. The red line in the plots indicate the methods which belong to the Pareto front, for example, these ones which are not dominated by the other methods. The following methods belong to the front: *ENN*, *HMN-E*, *GLVQ*(0.3), *LVQ2.1*(0.3), *HMN-EI*, *Drop2*, *Drop3* and *LVQ2.1*(0.1). Some other methods like *k-Means*(0.3) and *LVQ1* can be considered as the top ones because they lie very close to the Pareto front. From the top methods two algorithms provide compression less than 50%, these are *ENN* and *HMN-E*. These methods should be considered only when the compression is not the primary need.



(**a**)

(**b**)

Figure 11. Average results over all evaluated classifiers. Red line represents Pareto front. (**a**) Average performance ranks. (**b**) Average performance.

In theory, as indicated in the beginning of this article, the goal of data filtering methods is to keep the estimated conditional probabilities $P(c|\mathbf{x})$ unchanged before and after data filtering so that the $P(c|\mathbf{x})_\mathbf{T} = P(c|\mathbf{x})_\mathbf{P}$, but in reality each of these classifiers has its own probability estimation technique. So the one used by the decision trees which is based on the instance frequency calculation within the bin, do not match with the one of the nearest neighbor classifier. Moreover, SVM and Random Forest are more robust than *k*NN so they can better deal with the noisy samples than the data filters which internally use *k*NN to assess training instances.

The obtained results indicate that the size of the dataset matters. In general applying any of the examined data filters result in the decrease in accuracy, and a huge drop in prediction performance can be observed between compression 70% and 90%, so that the compression 70% can be considered as a kind of threshold below which we should not compress the dataset. Although, we also observed, that for bigger datasets instance selection methods proved more efficient allowing for higher compression. Interestingly, on average the prediction performance slowly drops even for the noise filters. The exception are 1-NN and Naive Bayes classifiers where some of the tested instance filtering methods (in particular *ENN* and the *LVQ* family) allowed to increase the accuracy. For the *k*NN with tuned *k* the accuracy may remain unchanged, so the benefit is the execution time of the prediction phase, which requires fewer distance calculations to make the decision.

The observed phenomenon can be interpreted taking into account that all of the tested instance selection methods were developed to work with *k*NN. As it was indicated in Section 2 instance selection methods can be considered wrappers for the kNN classifier, while for the remaining classifiers they work as filters. Some authors design specific algorithms for particular classifiers. The examples are the works of Kawulok and Nalepa who developed memetic [37] and genetic [38] algorithms for SVM, also de Mello and others developed an algorithm dedicated for the SVM [39]. In [7] we developed generalized *CNN* and *ENN* algorithms which work as wrappers in particular with MLP network. But these methods are strictly designed for given classifiers and can not be generalized so they were not considered in this research.

In the literature some authors use instance selection methods for balancing the data distribution of unbalanced classification problems [40]. In this scenario instance selection methods are applied to down-sample the majority class, and the minority classes remain unchanged, but this aspect was not considered in our study. Also the problem of applying instance selection methods to other tasks such as regression [41], multi-label learning [42] or stream mining [43] was not covered and requires further studies. Another open question which remains is deeper analysis of why particular of evaluated methods are better than the competitors. This requires independent analysis on lower number of methods and remains for future investigation.

In summery, when considering the use of initial data filtering for training set reduction one should consider *GLVQ*, *LVQ2.1* and in the case where it is needed to use the original training samples and not newly constructed prototypes one should consider *Drop2 Drop3* from the set of evaluated methods. These methods provide significant dataset size reduction and in general allow to obtain the higher prediction accuracy in comparison to the other methods with similar compression, but by applying them we should expect a drop in prediction accuracy for classifiers other than *k*NN.

Author Contributions: Conceptualization, M.B.; methodology, M.B.; software, M.B.; validation, M.B., M.Kordos.; formal analysis, M.B.; investigation, M.B. and M.K.; resources, M.B.; data curation, M.B.; writing—original draft preparation, M.B.; writing—review and editing, M.B., M.K.; visualization, M.B. and M.K.; supervision, M.B.; project administration, M.B.; funding acquisition, M.B. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by Silesian University of Technology BK-204/2020/RM4

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SVM	Support Vector Machine
Random Forest	Random Forest
*k*NN	k-Nearest Neighbor
1-NN	1-Nearest Neighbor
C4.5	C4.5 decision tree
GLM	Generalized Linear Model
Naive Bayes	Naive Bayes
MLP	Multi Layer Perceptron
Linear Regression	Linear Regression
ENN	Edited Nearest Neighbor Rule
CNN	Condenced Nearest Neighbor Rule
RENN	Repeated *ENN*
ICF	Iterative Case Filtering
IB3	Instance Based Learining version 3
IB2	Instance Based Learining version 2
GGE	Gabriel Graph Editing

RNGE	Relative Neighbor Graph Editing
MSS	Modified Selective Subset Selection
LVQ	Learning Vector Quantization
LVQ1	Learning Vector Quantization version 1
LVQ2	Learning Vector Quantization version 2
LVQ2.1	Learning Vector Quantization version 2.1
LVQ3	Learning Vector Quantization version 3
OLVQ1	Optimized Learning Vector Quantization
GLVQ	Generalized Learning Vector Quantization
SNG	Suppervised Neural Gas
CCIS	Class Conditional Instance Selection
HMN	Hit Miss Network
HMN-C	Hit Miss Network Condensation
HMN-E	Hit Miss Network Editing
HMN-EI	Hit Miss Network Iterative Editing
RNN	Reduced Nearest Neighbor Rule

References

1. Blachnik, M. Reducing Time Complexity of SVM Model by LVQ Data Compression. In *Artificial Intelligence and Soft Computing*; LNCS 9119; Springer: Berlin, Germany, 2015; pp. 687–695.
2. Duch, W.; Grudziński, K. Prototype based rules—New way to understand the data. In Proceedings of the IEEE International Joint Conference on Neural Networks, Washington, DC, USA, 15 July 2001; pp. 1858–1863.
3. Blachnik, M.; Duch, W. LVQ algorithm with instance weighting for generation of prototype-based rules. *Neural Networks* **2011**, *24*, 824–830. [CrossRef]
4. Kraskov, A.; Stögbauer, H.; Grassberger, P. Estimating mutual information. *Phys. Rev. E* **2004**, *69*, 066138. [CrossRef]
5. García, S.; Luengo, J.; Herrera, F. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowl. Based Syst.* **2016**, *98*, 1–29. [CrossRef]
6. Blachnik, M.; Kordos, M.; Wieczorek, T.; Golak, S. Selecting Representative Prototypes for Prediction the Oxygen Activity in Electric Arc Furnace. *LNCS* **2012**, *7268*, 539–547.
7. Kordos, M.; Blachnik, M.; Białka, S. Instance Selection in Logical Rule Extraction for Regression Problems. *LNAI* **2013**, *7895*, 167–175.
8. Abdulali, A.; Hassan, W.; Jeon, S. Stimuli-magnitude-adaptive sample selection for data-driven haptic modeling. *Entropy* **2016**, *18*, 222. [CrossRef]
9. Blachnik, M. Instance Selection for Classifier Performance Estimation in Meta Learning. *Entropy* **2017**, *19*, 583. [CrossRef]
10. Grochowski, M.; Jankowski, N. Comparison of Instance Selection Algorithms. II. Results and Comments. *LNCS* **2004**, *3070*, 580–585.
11. Borovicka, T.; Jirina, M., Jr.; Kordik, P.; Jirina, M. Selecting representative data sets. In *Advances in Data Mining Knowledge Discovery and Applications*; IntechOpen: London, UK, 2012.
12. Blachnik, M.; Duch, W. Prototype-based threshold rules. *Lect. Notes Comput. Sci.* **2006**, *4234*, 1028–1037.
13. García, S.; Derrac, J.; Cano, J.R.; Herrera, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435. [CrossRef]
14. Triguero, I.; Derrac, J.; Garcia, S.; Herrera, F. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Trans. Syst. Man, Cybern.* **2012**, *42*, 86–100. [CrossRef]
15. Hart, P. The condensed nearest neighbor rule. *IEEE Trans. Inf. Theory* **1968**, *16*, 515–516. [CrossRef]
16. Aha, D.; Kibler, D.; Albert, M. Instance-Based Learning Algorithms. *Mach. Learn.* **1991**, *6*, 37–66. [CrossRef]
17. Marchiori, E. Hit miss networks with applications to instance selection. *J. Mach. Learn. Res.* **2008**, *9*, 997–1017.
18. Barandela, R.; Ferri, F.J.; Sánchez, J.S. Decision boundary preserving prototype selection for nearest neighbor classification. *Int. J. Pattern Recognit. Artif. Intell.* **2005**, *19*, 787–806. [CrossRef]
19. Wilson, D.; Martinez, T. Reduction techniques for instance-based learning algorithms. *Mach. Learn.* **2000**, *38*, 257–268. [CrossRef]
20. Tomek, I. An experiment with the edited nearest-neighbor rule. *IEEE Trans. Syst. Man Cybern.* **1976**, *6*, 448–452.

21. Wilson, D. Assymptotic properties of nearest neighbour rules using edited data. *IEEE Trans. Syst. Man Cybern.* **1972**, *SMC-2*, 408–421. [CrossRef]
22. Sánchez, J.S.; Pla, F.; Ferri, F.J. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognit. Lett.* **1997**, *18*, 507–513. [CrossRef]
23. Brighton, H.; Mellish, C. Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.* **2002**, *6*, 153–172. [CrossRef]
24. Marchiori, E. Class conditional nearest neighbor for large margin instance selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 364–370. [CrossRef] [PubMed]
25. Nova, D.; Estévez, P.A. A review of learning vector quantization classifiers. *Neural Comput. Appl.* **2014**, *25*, 511–524. [CrossRef]
26. Blachnik, M.; Kordos, M. Simplifying SVM with Weighted LVQ Algorithm. *LNCS* **2011**, *6936*, 212–219.
27. Kordos, M.; Blachnik, M. Instance Selection with Neural Networks for Regression Problems. *LNCS* **2012**, *7553*, 263–270.
28. Arnaiz-González, Á.; Díez-Pastor, J.F.; Rodríguez, J.J.; García-Osorio, C. Instance selection of linear complexity for big data. *Knowl.-Based Syst.* **2016**, *107*, 83–95. [CrossRef]
29. De Haro-García, A.; Cerruela-García, G.; García-Pedrajas, N. Instance selection based on boosting for instance-based learners. *Pattern Recognit.* **2019**, *96*, 106959. [CrossRef]
30. Arnaiz-González, Á.; González-Rogel, A.; Díez-Pastor, J.F.; López-Nozal, C. MR-DIS: Democratic instance selection for big data by MapReduce. *Prog. Artif. Intell.* **2017**, *6*, 211–219. [CrossRef]
31. Blachnik, M.; Duch, W.; Wieczorek, T. Selection of prototypes rules – context searching via clustering. *LNCS* **2006**, *4029*, 573–582.
32. Kuncheva, L.; Bezdek, J. Presupervised and postsupervised prototype classifier design. *IEEE Trans. Neural Networks* **1999**, *10*, 1142–1152. [CrossRef]
33. Herrera, F. KEEL, Knowledge Extraction based on Evolutionary Learning. Spanish National Projects TIC2002-04036-C05, TIN2005-08386-C05 and TIN2008-06681-C06. 2005. Available online: http://www.keel.es (accessed on 1 May 2020).
34. Blachnik, M.; Kordos, M. Information Selection and Data Compression RapidMiner Library. In *Machine Intelligence and Big Data in Industry*; Springer: Berlin, Germany, 2016; pp. 135–145.
35. Alcalá-Fdez, J.; Fernández, A.; Luengo, J.; Derrac, J.; García, S.; Sanchez, L.; Herrera, F. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Log. Soft Comput.* **2011**, *17*, 255–287.
36. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
37. Nalepa, J.; Kawulok, M. Adaptive memetic algorithm enhanced with data geometry analysis to select training data for SVMs. *Neurocomputing* **2016**, *185*, 113–132. [CrossRef]
38. Kawulok, M.; Nalepa, J. Support vector machines training data selection using a genetic algorithm. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*; Springer: Berlin, Germany, 2012; pp. 557–565.
39. de Mello, A.R.; Stemmer, M.R.; Barbosa, F.G.O. Support vector candidates selection via Delaunay graph and convex-hull for large and high-dimensional datasets. *Pattern Recognit. Lett.* **2018**, *116*, 43–49. [CrossRef]
40. Devi, D.; Purkayastha, B. Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance. *Pattern Recognit. Lett.* **2017**, *93*, 3–12. [CrossRef]
41. Arnaiz-González, Á.; Díez-Pastor, J.; Rodríguez, J.J.; García-Osorio, C.I. Instance selection for regression by discretization. *Expert Syst. Appl.* **2016**, *54*, 340–350, doi:10.1016/j.eswa.2015.12.046. [CrossRef]
42. Kordos, M.; Arnaiz-González, Á.; García-Osorio, C. Evolutionary prototype selection for multi-output regression. *Neurocomputing* **2019**, *358*, 309–320. [CrossRef]
43. Gunn, I.A.; Arnaiz-González, Á.; Kuncheva, L.I. A Taxonomic Look at Instance-based Stream Classifiers. *Neurocomputing* **2018**, 286, 167–178. [CrossRef]

applied sciences

MDPI

Article

Selection of Support Vector Candidates Using Relative Support Distance for Sustainability in Large-Scale Support Vector Machines

Minho Ryu [1,2] and Kichun Lee [2,*]

[1] Vision AI Labs, SK Telecom, Seoul 04539, Korea; ryumin93@sktbrain.com

[2] Department of Industrial Engineering, College of Engineering, Hanyang University, Seoul 04763, Korea

* Correspondence: skylee@hanyang.ac.kr; Tel.: +82-02-2220-0478

Received: 8 September 2020; Accepted: 29 September 2020; Published: 6 October 2020

Abstract: Support vector machines (SVMs) are a well-known classifier due to their superior classification performance. They are defined by a hyperplane, which separates two classes with the largest margin. In the computation of the hyperplane, however, it is necessary to solve a quadratic programming problem. The storage cost of a quadratic programming problem grows with the square of the number of training sample points, and the time complexity is proportional to the cube of the number in general. Thus, it is worth studying how to reduce the training time of SVMs without compromising the performance to prepare for sustainability in large-scale SVM problems. In this paper, we proposed a novel data reduction method for reducing the training time by combining decision trees and relative support distance. We applied a new concept, relative support distance, to select good support vector candidates in each partition generated by the decision trees. The selected support vector candidates improved the training speed for large-scale SVM problems. In experiments, we demonstrated that our approach significantly reduced the training time while maintaining good classification performance in comparison with existing approaches.

Keywords: support vector machine; decision tree; large-scale dataset; relative support distance; support vector candidates

1. Introduction

Support vector machines (SVMs) [1] have been a very powerful machine learning algorithm developed for classification problems, which works by recognizing patterns via kernel tricks [2]. Because of its high performance and great generalization ability compared with other classification methods, the SVM method is widely used in bioinformatics, text and image recognition, and finances, to name a few. Basically, the method finds a linear boundary (hyperplane) that represents the largest margin between two classes (labels) in the input space [3–6]. It can be applied to not only linear separation but also nonlinear separation using kernel functions. Its nonlinear separation can be achieved via kernel functions, which map the input space to a high-dimensional space, called feature space where optimal separating hyperplane is determined in the feature space. In addition, the hyperplane in the feature space, which achieves a better separation of training data, is translated to a nonlinear boundary in the original space [7,8]. The kernel trick is used to associate the kernel function with the mapping function, bringing forth a nonlinear separation in the input space.

Due to the growing speed of data acquisition on various domains and the continual popularity of SVMs, large-scale SVM problems frequently arise: human detection using histogram of oriented gradients by SVMs, large-scale image classification by SVMs, disease classification using mass spectrum by SVMs, and so forth. Even though SVMs show superior classification performance, their computing time and storage requirements increase dramatically with the number of instances, which is a major obstacle [9,10]. As the goal of SVMs is to find the optimal separating hyperplane that maximizes the margin between two classes, they should solve a quadratic programming problem. In practice, the time complexity in the training phase of the SVM method is at least $O\left(n^2\right)$, where n is the number of data samples, depending on the kernel function [11]. Indeed, several approaches have been applied to improve the training speed of SVMs. Sequential minimal optimization (SMO) [12], SVM-light [13], simple support vector machine (SSVM) [14] and library of support vector machine (LibSVM) [15] are among others. Basically, they break the problem into a series of small problems that can be easily solved, reducing the required memory size.

Additionally, data reduction or selection methods have been introduced for large-scale SVM problems. Reduced support vector machines (RSVMs) are a random sampling method that, being quite simple, uses a small portion of the large dataset [16]. However, it needs to be applied several times and unimportant observations are equally sampled. The method presented by Collobert et al. efficiently parallelizes sub-problems, fitting to very large-size SVM problems [17]. It used cascades of SVMs in which data are split into subsets to be optimized separately with multiple SVMs instead of analyzing the whole dataset. A method based on the selection of candidate vectors (CVS) was presented using relative pair-wise Euclidean distances in the input space to find the candidate vectors in advance [18]. Because the only selected samples are used in the training phase, it shows fast training speed. However, its classification performance is relatively worse than that of the conventional SVM, and the need for selecting good candidate vectors arise.

Besides, for large-scale SVM problems, a joint approach that combines SVM with other machine learning methods has emerged. Many evolutionary algorithms have been proposed to select training data for SVMs [19–22]. Although they have shown promising results, these methods need to be executed multiple times to decide proper parameters and training data, which is computationally expensive. Decision tree methods also have been commonly proposed to reduce training data because the training time is proportional to $O\left(np^2\right)$ where p represents discrete input variables [23] so is faster than traditional SVMs. The decision tree method recursively decomposes the input data set into binary subsets through independent variables when the splitting condition is met. In supervised learning, decision trees, bringing forth random forests, are one of the most popular models because they are easy to interpret and computationally inexpensive. Indeed, taking advantage of decision trees, several researches combining SVMs with decision trees have been proposed for large-size SVM problems. Fu Chang et al. [24] presented a method that uses a binary tree to decompose an input data space into several regions and trains an SVM classifier on each of the decomposed regions. Another method using decision trees and Fisher's linear discriminant was also proposed for large-size SVM problems in which they applied Fisher's linear discriminant to detect 'good' data samples near the support vectors [25]. Cervantes et al. [26] also utilized a decision tree to select candidate support vectors using the support vectors annotated by SVM trained by a small portion of training data. Their approaches, however, are limited in that it cannot properly handle the regions that have nonlinear relationships.

The ultimate aim in dealing with large-scale SVM problems is to reduce the training time and memory consumption of SVMs without compromising the performance. For this goal, it would be worth finding good support vector candidates as a data-reduction method. Thus, in this paper we present a method that finds support vector candidates based on decision trees that works better than previous methods. We determine the decision hyperplane using support vector candidates chosen among the training dataset. In this proposed approach, we introduce a new concept, relative support distance, to effectively find candidates using decision trees in consideration of nonlinear relationships between local observations and labels. Decision tree learning decomposes the input space and helps find subspaces of the data where the majority class labels are opposite to each other. Relative support distance measures a degree that an observation is likely to be a support vector, using a virtual hyperplane that bisects the two centroids of two classes and the nonlinear relationship between the hyperplane and each of the two centroids.

This paper is organized as follows. Section 2 provides the overview of SVMs and decision trees that are exploited in our algorithm. In Section 3, we introduce the proposed method of selecting support vector candidates using relative support distance measures. Then, in Section 4, we provide the results of experiments to compare the performance of the proposed method with that of some existing methods. Lastly, in Section 5, we conclude this paper with future research directions.

2. Preliminaries

In this section, we briefly summarize the concepts of support vector machines and decision trees. Relating to the concepts, we then introduce the concept of relative support distance to measure the possibility of being a support vector in training data.

2.1. Support Vector Machines

Support vector machines (SVMs) [1] are generally used for binary classification. Given n pairs of instances with input vectors $\{\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \ldots,\ \boldsymbol{x}_n\}$ and response variables $\{y_1, y_2, \ldots, y_n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1,\ 1\}$, SVMs present a decision function in a hyperplane that optimally separates two classes:

$$y = sign(\boldsymbol{w}^t\boldsymbol{x} + b) \tag{1}$$

where $\boldsymbol{w}$ is a weight vector and b is a bias term. The margin is the distance between the hyperplane and the training data nearest the hyperplane. The distance from an observation to the hyperplane is given by $\left|d(\boldsymbol{x})\right|/\boldsymbol{w}$. To find the hyperplane that maximizes the margin, we solve the problem by transforming it to its dual problem, introducing the Lagrange multipliers. Namely, in soft-margin SVMs with penalty parameter C, we find $\boldsymbol{w}$ by the following optimization problem:

$$\begin{gathered} max \sum_i^n \alpha_i - \frac{1}{2}\sum_i^n\sum_j^n \alpha_i\alpha_j y_i y_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \\ subject\ to \quad \sum_i^n \alpha_i y_i = 0, \\ 0 \le \alpha_i \le C/n, i = 1, \ldots, n. \end{gathered} \tag{2}$$

where $C > 0$, α_i, $i = 1, \ldots, n$, are the dual variables corresponding $\boldsymbol{x}_i$, and all the $\boldsymbol{x}_i$ corresponding to nonzero α_i are called support vectors. By numerically solving the problem (2) for α_i, we obtain α_i^* and compute $\boldsymbol{w}^* = \sum_i \alpha_i^* y_i \boldsymbol{x}_i$ and $b^* = y_i - \boldsymbol{w}^*\boldsymbol{x}_i$ for $0 < \alpha_i^* < C/n$. The kernel function $K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ is the inner product of the mapping function: $K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \phi(\boldsymbol{x}_i)^T\phi\left(\boldsymbol{x}_j\right)$. The mapping function $\phi(\boldsymbol{x})$ maps the input vectors to high-dimensional feature spaces. Well-known kernel functions are polynomial kernels, tangent kernels, and radial basis kernels. In this research, we chose the radial basis kernel function (RBF) with a free parameter γ denoted as

$$K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right) \tag{3}$$

Notice that the radial basis kernel, possessing the mapping function $\phi(\boldsymbol{x})$ with an infinite number of dimensions [27], is flexible and the most widely chosen.

2.2. Decision Tree

A decision tree is a general tool in data mining and machine learning used as a classification or regression model in which a tree-like graph of decisions is formed. Among the well-known algorithms such as CHAID (chi-squared automatic interaction detection) [28], CART (classification and regression tree) [29], C4.5 [30], and QUEST (quick, unbiased, efficient, statistical tree) [31], we use CART which is very similar to C4.5 since it uses a binary splitting criterion applied recursively and leaving no empty leaf. Decision tree learning builds its model based on recursive partitioning of training data into pure or homogeneous sub-regions. Prediction process of classification or regression can be expressed by inference rules based on the tree structure of the built model, so it can be interpreted and understood easier than other methods. The tree building procedure begins at the root node, which includes all instances in the training data. To find the best possible variable to split the node into two child nodes, we check all possible splitting variables (called splitters), as well as all possible values of the variable used to split the node. It involves an $O(pn \log n)$ time complexity where p is the number of input variables and n is the size of the training data set [32]. In choosing the best splitter, we can use some impurity metrics such as entropy or Gini impurity. For example, the Gini impurity function: $im(T) = 1 - \sum_y p(T = y)^2$, where $p(T = y)$ is the proportion of observations where class type T is y. Next, we define the difference between the weighted impurity measure of the parent node and the two child nodes. Let us denote the impurity measure of the parent node by $im(T)$; the impurity measures of the two child nodes by $im\left(T_{left}\right)$ and $im\left(T_{right}\right)$; the number of parent node instances by X_T; and the number of the child node instances by $X_{T,left}$ and $X_{T,right}$. We choose the best splitter by the query that decreases the impurity as much as possible:

$$\Delta im(T) = im(T) - \frac{X_{T,left}}{X_T} im\left(T_{left}\right) - \frac{X_{T,right}}{X_T} im\left(T_{right}\right). \tag{4}$$

There are two methods called pre-pruning and post-pruning to avoid over-fitting in decision tree. The pre-pruning method uses stopping conditions before over-fitting occurs. It attempts to stop separating each node if specified conditions are met. The latter method makes a tree over-fitting and determines an appropriate tree size by backward pruning of the over-fitted tree [33]. Generally, the post-pruning is known as more effective than the pre-pruning. Therefore, we use the post-pruning algorithm.

3. Tree-Based Relative Support Distance

In order to cope with large-scale SVM problems, we propose a novel selection method for support vector candidates using a combination of tree decomposition and relative support distance. We aim to reduce the training time of SVMs for the numerical computation of α_i in (2) which produces w^* and b^* in (1) by selecting good support vectors in advance that are a small subset of the training data. To illustrate our concept, we start with a simple example in Figure 1, where the distribution of the iris data is shown: for the details of the data, refer to Fisher [34]. In short, the iris dataset describes iris plants using four continuous features. The data set contains 3 classes of 50 instances as Iris Setosa, Iris Versicolor, or Iris Virginica. We decompose the input space into several regions by decision tree learning. After training an SVM model for the whole dataset, we mark support vectors by filled shapes. Each region has its own majority class label, and the boundaries are between the two majority classes. The support vectors are close to the boundaries. In addition, we notice that they are located relatively far away from the center of the data points with the majority class label in a region.

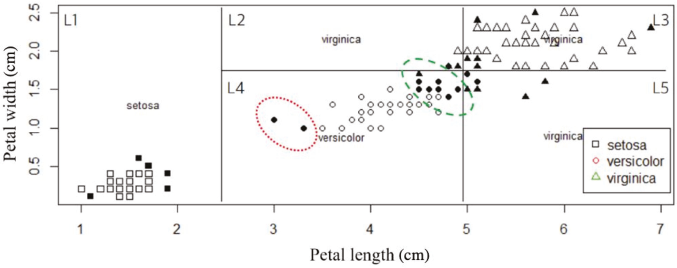


Figure 1. The construction of a decision tree and SVMs for the iris data shows the boundaries and support vectors (the filled shapes). The support vectors in the regions are located far away from the majority-class centroid.

In light of this, we describe our algorithm to find a subset of support vectors that determine the separating hyperplane. We divide a training dataset into several decomposed regions in the input space by decision tree learning. This process brings each decomposed region to have most of the data points with the majority class label by the tree learning algorithm. Next, we detect adjacent regions in which the majority class is opposite to that of each region. We define this kind of region as distinct adjacent region. Then we calculate a new distance measure, relative support distance, with the data points in the selected region pairs. The procedure of the algorithm is as follows:

1. Decompose the input space by decision tree learning.
2. Find distinct adjacent regions, which mean adjacent regions whose majority class is different from that of each region.
3. Calculate the relative support distances for the data points in the found distinct adjacent regions.
4. Select the candidates of support vectors according to the relative support distances.

3.1. Distinct Adjacent Regions

After applying decision tree learning to the training data, we detect adjacent regions. The decision tree partitions the input space into several leaves (also denoted by terminal nodes) by reducing some impurity measures such as entropy. Following the approach of detecting adjacent regions introduced by Chau [25], we put in mathematical conditions for being adjacent regions and relate it to the relative support distance. Firstly, we represent each terminal node of a learned decision tree as follows:

$$L_q = \bigcap_{j=1}^{p} b_{qj}, \quad l_{qj} \leq b_{qj} \leq h_{qj}, \tag{5}$$

where L_q is the qth leaf in the tree structure and b_{qj} is the boundary range for the jth variable of the qth leaf with its lower bound l_{qj} and upper bound h_{qj}. Recall that p is the number of input variables. We should check whether each pair of leaves, L_o and L_q, meet the following criteria:

$$h_{os} = l_{qs} \;\; or \;\; l_{os} = h_{qs}, \tag{6}$$

$$l_{qk} \leq l_{ok} \leq h_{qk} \;\; or \;\; l_{qk} \leq h_{ok} \leq h_{qk}, \tag{7}$$

where s and k are one of the input variables, $1 \leq s \leq p$, $1 \leq k \leq p$, and $s \neq k$. That is to say, if two leaves L_o and L_q are adjacent regions, they have to share one variable, represented by the variable s in Equation (6), and one boundary, induced by the variable k in (7). Among all adjacent regions, we only consider distinct adjacent regions. For example, in Figure 2, the neighbors of L_1 are L_2, L_4, and L_5: $\{L_1, L_5\}$, however, does not form an adjacent region pair. $\{L_3, L_5\}$ is an adjacent region pair but not distinct since those regions have the same majority class. Therefore, the distinct adjacent regions in the example are only $\{L_1, L_2\}$, $\{L_1, L_4\}$, $\{L_2, L_3\}$, $\{L_2, L_5\}$, and $\{L_4, L_5\}$. Distinct adjacent regions are summarized in Table 1. Now, we apply the measure of relative support distance to select support vector candidates in the found distinct adjacent regions for each region.

Table 1. Partition of input regions and distinct adjacent regions.

Region	Distinct Adjacent Regions
L_1	L_2, L_4
L_2	L_1, L_3, L_5
L_3	L_2
L_4	L_1, L_5
L_5	L_2, L_4

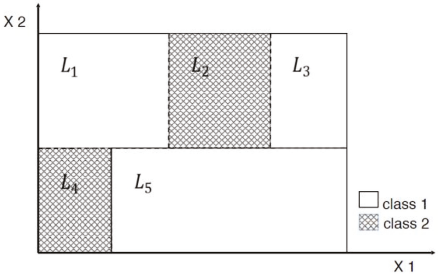


Figure 2. Distinct adjacent regions are $\{L_1, L_2\}$, $\{L_1, L_4\}$, $\{L_2, L_3\}$, $\{L_2, L_5\}$, and $\{L_4, L_5\}$.

3.2. Relative Support Distance

Support vectors (SVs) play a substantial role in determining the decision hyperplane in contrast to non-SV data points. We extract data points in the training data that are most likely to be the support vectors, constructing a set of support vector candidates. Given two distinct adjacent regions L_1 and L_2 from the previous step, let us assume the majority class label of L_1 is $y = 1$ and that of L_2, $y = 2$ without loss of generality. First, we calculate the centroid ($\boldsymbol{m}_c$) for each majority class label as follows: for an index set $S_c = \{i|\boldsymbol{x}_i \in L_c$ and the label of $\boldsymbol{x}_i = c\}$,

$$\boldsymbol{m}_c = \frac{1}{n_c}\sum_{i \in S_c} \boldsymbol{x}_i, \tag{8}$$

where $c \in \{1,\ 2\}$ and n_c is the cardinality of index set S_c.

In other words, $\boldsymbol{m}_c$ is the majority-class centroid of data points in L_c, of which the labels are $y = c$. Next, we create a virtual hyperplane that bisects the line from $\boldsymbol{m}_1$ to $\boldsymbol{m}_2$:

$$\begin{aligned} \boldsymbol{M} &= \tfrac{1}{2}(\boldsymbol{m}_1 + \boldsymbol{m}_2), \\ \boldsymbol{W} &= \boldsymbol{m}_1 - \boldsymbol{m}_2, \end{aligned} \tag{9}$$

where M is the middle point of the two majority-class centroids. The virtual hyperplane is given by $H(\boldsymbol{x}) = 0$, where

$$H(\boldsymbol{x}) = \boldsymbol{W}^t(\boldsymbol{x} - \boldsymbol{M}). \tag{10}$$

Lastly, we calculate the distance r_x between each data point x in S_c and $\boldsymbol{m}_c$ and the distance h between each data point in S_c and the virtual hyperplane $H(\boldsymbol{x}) = 0$:

$$\begin{aligned} r_{x_{c,l}} &= \left\|\boldsymbol{x}_{c,l} - \boldsymbol{m}_c\right\|, \\ h_{x_{c,l}} &= \frac{|H(x_{c,l})|}{\|W\|} = \frac{W^t(x_{c,l} - M)}{\|m_1 - m_2\|}, \end{aligned} \tag{11}$$

where $\boldsymbol{x}_{c,l}$ is the lth data point belonging to S_c. Figure 3 shows a conceptual description of r and h using the virtual hyperplane in a leaf. After calculating r_x and h_x, we apply feature scaling to bring all values into the range between 0 and 1. Our observation is that data points lying close to the virtual hyperplane are likely to be support vectors. In addition, data points lying close to the centroid are less likely to be support vectors. In light of these observations, we select data points lying near the hyperplane and far away from the centroid. For this purpose, we define the relative support distance $T(r_x, h_x)$ as follows:

$$T(r_x, h_x) = \frac{1}{(1 + e^{-r_x})h_x}. \tag{12}$$

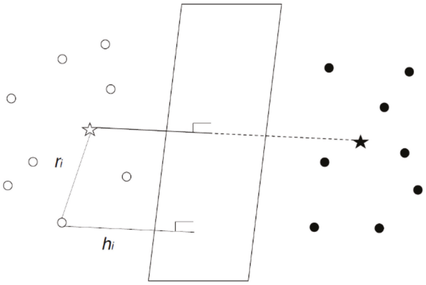
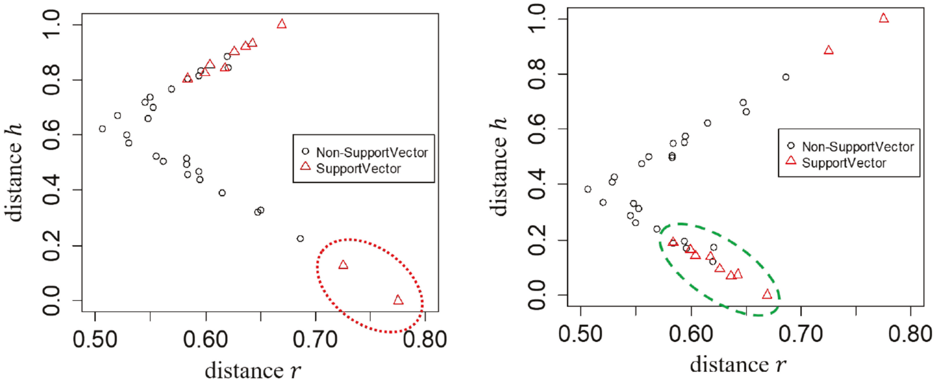


Figure 3. Distances r from the center and h from the virtual hyperplane are shown. The two-star shapes are the centroid of each class.

The larger $T(r_x, h_x)$ becomes, the more likely that the associated $\boldsymbol{x}$ is a support vector.

The relationship between support vectors and distances r and h is illustrated in Figure 4. We use leaves L_4 with distinct adjacent regions L_1 and L_2 in Figure 1. In Figure 4, the observations marked by circles are non-support vectors while those by triangles (in red) are support vectors selected after training all data by SVMs. The distances r and h of L_4 relative to L_1 are in Figure 4a, and the relative support distance measures in Figure 4c. Likewise, those of L_4 relative to L_2 are in Figure 4b,d. We observe that the observations, marked by triangles and surrounded by a red ellipsoid in Figure 1, correspond to the support vectors surrounded by a red ellipsoid in region L_4 in Figure 4a, and they have larger values of relative support distance as shown in Figure 4c. Similarly, we notice that the observations, marked by triangles and surrounded by a green ellipsoid in Figure 4b, correspond to the support vectors surrounded by a green ellipsoid in region L_4 in Figure 1, and they also have larger values of relative support distance as shown in Figure 4d. The support vectors in L_4 are obtained by collecting observations with large values of relative support distance, for example by the rule $T(r_x, h_x) > 0.9$, from both the pair of L_4 and L_1 and the pair of L_4 and L_2. The results reveal that the observations that have a mostly larger distance r and shorter distance h are likely to be support vectors.



(**a**) (**b**)

Figure 4. *Cont.*

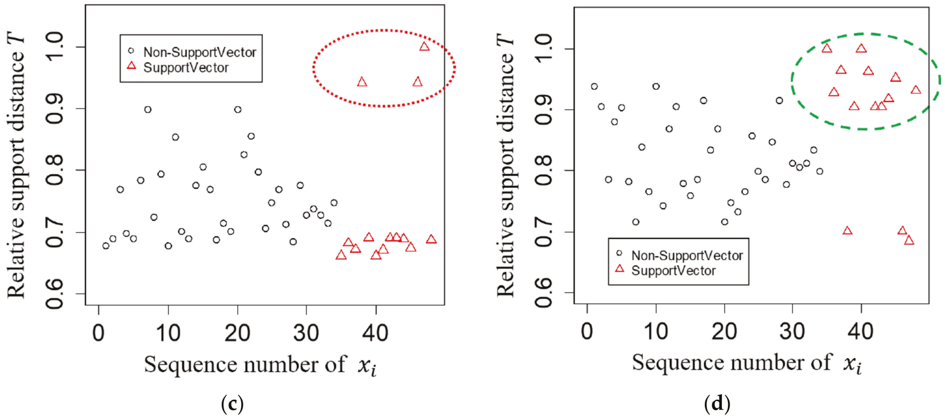


Figure 4. Illustration of the distances r and h according to distinct adjacent regions for the iris data in Figure 1. (**a**) Distances for leaf L_4 relative to L_1 are shown. Notice the support vectors captured by leaf L_4 to leaf L_1 are in the (dotted) red ellipsoid. (**b**) Distances for leaf L_4 relative to L_2 are shown. Notice the support vectors captured by leaf L_4 relative to leaf L_2 are in the (dashed) green ellipsoid. (**c**) Relative support distances of the observations x_i in leaf L_4 relative to L_1 are shown. Notice the support vectors captured by leaf L_4 to leaf L_1 are in the (dotted) red ellipsoid. (**d**) Relative support distances of the observations x_i in leaf L_4 relative to L_2 are shown. Notice the support vectors captured by leaf L_4 relative to leaf L_2 are in the (dashed) green ellipsoid.

For each region, we calculate pairwise relative support distance with distinct adjacent regions and select a fraction of the observations, denoted by parameter β, in the decreasing order by $T(r_x, h_x)$ as a candidate set of support vectors. That is to say, for each region, we select the top β fraction of training data based on $T(r_x, h_x)$. Parameter β represents the proportion of the selected data points, between 0 and 1. For example, when β is set to 1, all data points are included in the training of SVMs. When $\beta = 0.1$, we exclude 90% of the data points and reduce the training data set to 10%. Finally, we combine relative support distance with random sampling, which means that a half of the training candidates are selected based on the proposed distance and the others are selected by random sampling. Though being quite informative for selecting possible support vectors, the proposed distance is calculated locally with distinct adjacent regions. Therefore, random sampling can compensate this property by providing whole data distribution information.

4. Experimental Results

In the experiments, we compare the proposed method, tree-based relative support distance (denoted by TRSD), with some previously suggested methods, specifically SVMs with candidate vectors selection, denoted by CVS [18], and SVM with Fisher linear discriminant analysis, denoted by FLD [25], as well as standard SVMs, denoted by SVM. For all comparing methods, we use LibSVM [15] since it is one of the fastest methods for training SVMs. The experiments are run on a computer with the following features: Core i5 3.4 GHz processor, 16.0 GB RAM, Windows 10 enterprise operating system. The algorithms are implemented in the R programming language. We use 18 datasets which are from UCI Machine Learning Repository [35] and LibSVM Data Repository [36] except the checkerboard dataset [37]: a9a, banana, breast cancer, four-class, German credit, IJCNN-1 [38], iris, mushroom, phishing, Cod-RNA, skin segmentation, waveform, and w8a. Iris and Waveform datasets are modified for binary classification problems by assigning one class to positive and the others to negative. Table 2 shows a summary of the datasets used in the experiments where Size is the number of instances in dataset and Dim is the number of features.

Table 2. Datasets for experiments.

Dataset	Size	Dim	$\lvert y_i = +1 \rvert$	$\lvert y_i = -1 \rvert$
Iris-Setosa	150	4	50	100
Iris-Versicolor	150	4	50	100
Iris-Virginia	150	4	50	100
Breast Cancer	683	10	444	239
Four-class	862	2	555	307
Checkerboard	1000	2	514	486
German Credit	1000	24	700	300
Waveform-0	5000	21	1657	3343
Waveform-1	5000	21	1657	3343
Waveform-2	5000	21	1657	3343
Banana	5300	2	2924	2376
Mushroom	8124	112	3916	4208
Phishing	11,055	68	4898	6157
w8a	45,546	300	44,226	1320
a9a	48,842	123	37,155	11,687
IJCNN-1	141,691	22	128,126	13,565
Skin Segmentation	245,057	3	50,859	194,198
Cod-RNA	488,565	8	325,710	162,855

For testing, we apply three-fold cross validation, repeated three-times, by shuffling each dataset and dividing it into three parts, and use two parts as the training dataset, the other part as a testing dataset with different seeds. We use the RBF kernel for training SVMs in all tested methods. For each experiment, cross validation and grid search are used for tuning two hyper-parameters: the penalty factor C and the RBF kernel parameter γ in Equation (3). Hyper-parameters are searched by a two-dimensional grid with C $\in$ {0.1, 1, 10, 100} and $\gamma \in$ {0.001, 0.01, 0.1, 1, $1/p$} where p is the number of features. Table 3 shows the values used for each dataset in the experiments. Moreover, we vary the fraction of data points in each region β from 0.1 to 0.3 with the interval of 0.1.

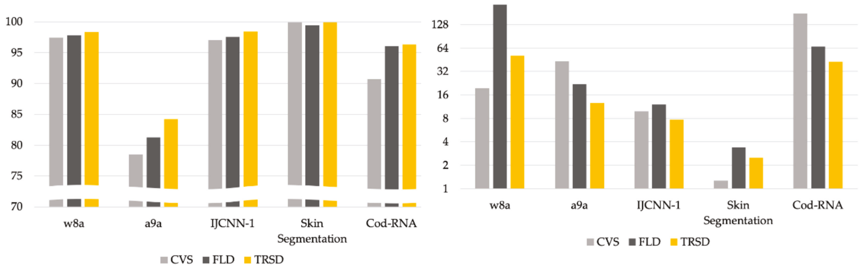
Table 3. Hyper-parameters setting for the experiments.

Method	Dataset	Penalty Factor C	RBF Kernel γ	Dataset	Penalty Factor C	RBF Kernel r
SVM	Iris-Setosa	10	0.001	Waveform-2	0.1	$1/p$
CVS		100	0.001		10	0.01
FLD		100	0.001		100	0.001
TRSD		100	0.001		1	$1/p$
SVM	Iris-Versicolor	10	0.1	Banana	1	1
CVS		100	0.1		100	1
FLD		0.1	0.01		10	$1/p$
TRSD		10	1		1	1
SVM	Iris-Virginia	100	0.01	Mushroom	100	0.001
CVS		100	0.01		10	0.001
FLD		100	0.001		0.1	0.01
TRSD		100	0.1		100	0.001
SVM	Breast Cancer	1	0.01	Phishing	10	$1/p$
CVS		10	0.001		1	0.01
FLD		100	0.001		100	0.001
TRSD		1	0.1		100	0.001
SVM	Four-class	10	1	w8a	10	0.001
CVS		100	1		10	0.01
FLD		100	1		1	0.001
TRSD		10	1		10	0.001
SVM	Checkerboard	100	1	a9a	10	0.001
CVS		100	1		10	0.01
FLD		100	1		100	0.001
TRSD		100	1		10	0.001

Table 3. *Cont.*

Method	Dataset	Penalty Factor C	RBF Kernel γ	Dataset	Penalty Factor C	RBF Kernel r
SVM	German Credit	100	0.001	IJCNN-1	10	0.1
CVS		1	$1/p$		100	$1/p$
FLD		0.1	0.001		100	0.01
TRSD		1	0.01		10	$1/p$
SVM	Waveform-0	1	0.01	Skin Segmentation	100	1
CVS		10	0.01		100	1
FLD		1	0.01		100	0.1
TRSD		1	0.01		100	1
SVM	Waveform-1	1	$1/p$	Cod-RNA	10	1
CVS		10	$1/p$		100	0.001
FLD		1	$1/p$		100	0.01
TRSD		1	$1/p$		10	$1/p$

We compare the performance of SVM, CVS, FLD, and TRSD in terms of classification accuracy and the training time (in seconds), summarized in Table 4. We also depicted the performance comparison of the proposed TRSD with CVS and FLD on the five largest datasets when $\beta = 0.1$ in Figure 5. We used log-2 scale for y-axis in Figure 5b. In Table 4, Acc is the accuracy on test data; σ is the standard deviation; and Time is the training time in seconds. Even though the accuracy of the proposed algorithm is slightly degraded in a few cases, it is higher than that of CVS and FLD in most cases. In addition, as β is greater, the accuracy of the proposed algorithm enhanced substantially. For small datasets, there is no significant improvement on computation time compared to the standard SVM since those datasets are already small enough. However, we notice that the training time of TRSD improved quite much when using the large-scale datasets.



(**a**) Accuracy

(**b**) Computational time

Figure 5. Comparison of the proposed tree-based relative support distance (TRSD) with candidate vectors (CVS) and Fisher linear discriminant analysis (FLD) on the five largest datasets.

For statistical analysis, we also performed Friedman test to see that there exists significant difference between the multiple comparing methods in terms of accuracy. If the null hypothesis of the Friedman test is rejected, we performed Dunn's test. Table 5 shows the summary of the Dunn's test results at the significant level $\alpha = 0.05$. In Table 5, the entries (1) TRSD $>$ CVS (FLD); (2) TRSD $\approx$ CVS (FLD); (3) TRSD $<$ CVS (FLD), respectively, denote that: (1) the performance of TRSD is significantly better than CVS (FLD); (2) there in no significant difference between the performances of TRSD and CVS (FLD); and (3) the performance of TRSD is significantly worse than CVS (FLD). Each number in Table 5 means the number of datasets. At $\beta = 0.1$, our proposed method is significantly better than CVS and FLD in 10 and 9 cases among 18 datasets. These numbers increase to 11 and 10 at $\beta = 0.3$. On the other hand, our proposed method is significantly worse than CVS only in 2, 3 and 3 cases; than FLD in 0, 1 and 0 cases at $\beta = 0.1,\ 0.2,\ 0.3$ respectively. Based on the observations in the experiments, we can conclude that our proposed method generates an effective reduction of the training datasets while producing better performance than the existing data reduction approaches.

Table 4. Comparisons in terms of accuracy and time on datasets (the results of support vector machines (SVM) are not bolded, because it had access to all examples).

Dataset	SVM			CVS			FLD			TRSD		
	acc	σ	Time	acc	σ	Time	Acc	σ	Time	Acc	σ	Time
$\beta = 0.1$												
Iris-Setosa	100	0	0.01	**100**	0	0.01	**100**	0	0.01	**100**	0	0.01
Iris-Versicolor	95.43	0.98	0.01	78.54	12.71	0.01	52.29	17.53	0.02	**87.71**	2.22	0.02
Iris-Virginia	96.57	2.24	0.01	87.99	6.55	0.01	**91.71**	4.07	0.02	91.5	4.72	0.01
Breast Cancer	96.99	0.89	0.01	95.36	0.5	0.01	**96.3**	0.95	0.08	96.24	1.04	0.04
Four-class	100	0	0.01	93.19	0.08	0.01	**93.28**	3.58	0.07	93.24	2.4	0.05
Checkerboard	94.3	0.63	0.03	**84.61**	1.47	0.01	-	-	-	79.34	3.13	0.08
German Credit	75.7	0.7	0.09	69.87	0.75	0.01	70	0.04	0.42	**70.55**	0.97	0.21
Waveform-0	89.85	0.63	1.03	85.5	0.73	0.21	87.51	0.65	0.58	**89.01**	0.51	0.34
Waveform-1	91.26	0.3	0.68	83.7	0.87	0.2	89.46	0.38	0.66	**90.06**	0.5	0.35
Waveform-2	91.76	0.48	0.94	84.52	1.23	0.21	89.77	0.97	0.74	**90.31**	0.48	0.38
Banana	90.6	0.69	0.59	64.13	2.69	0.03	83.46	1.77	0.09	**88.8**	0.4	0.06
Mushroom	100	0	1.47	90.67	0.62	2.08	84.99	7.23	1.11	**99.83**	0.12	0.79
Phishing	96.69	0.26	4.59	90	0.49	2.28	93.28	0.69	1.46	**93.96**	0.26	0.79
w8a	99.11	0.04	114.85	97.4	0.05	19.64	97.83	0.3	231.89	**98.38**	0.09	51.18
a9a	84.7	0.14	373.24	78.48	0.29	43.19	81.27	0.72	21.91	**84.24**	0.16	12.59
IJCNN-1	99.27	0.7	224.29	97.02	0.12	9.83	97.55	0.1	12	**98.39**	0.05	7.71
Skin Segmentation	99.94	0	20.23	**99.93**	0.01	1.27	99.45	0.11	3.39	99.89	0.01	2.51
Cod-RNA	96.98	0.03	4425.55	90.68	0.08	178.3	96.05	0.08	66.93	**96.32**	0.03	42.85
$\beta = 0.2$												
Iris-Setosa	100	0	0.01	**100**	0	0.01	**100**	0	0.01	**100**	0	0.01
Iris-Versicolor	95.43	0.98	0.01	89.44	3.92	0.01	51.72	17.02	0.02	**93.43**	1.48	0.02
Iris-Virginia	96.57	2.24	0.01	93.43	2.2	0.01	94.26	2.17	0.02	90.31	3.03	0.02
Breast Cancer	96.99	0.89	0.01	95.55	1.09	0.01	96.49	0.91	0.07	96.3	0.95	0.06
Four-class	100	0	0.01	96.42	1.78	0.01	95.38	1.63	0.07	**98.81**	0.8	0.05
Checkerboard	94.3	0.63	0.03	**93.66**	1.09	0.01	-	-	-	83.66	2.48	0.08
German Credit	75.7	0.7	0.09	70.04	0.01	0.02	70	0.04	0.35	**70.56**	0.62	0.25
Waveform-0	89.85	0.63	1.03	88.35	0.37	0.27	88.55	0.73	0.76	**89.28**	0.73	0.54
Waveform-1	91.26	0.3	0.68	86.45	0.56	0.26	89.89	0.37	0.78	**90.52**	0.49	0.5
Waveform-2	91.76	0.48	0.94	88.21	0.5	0.27	90.39	0.54	0.99	**90.62**	0.58	0.51
Banana	90.6	0.69	0.59	79.86	2.02	0.1	85.14	1.5	0.1	**89.69**	0.36	0.07
Mushroom	100	0	1.47	97.09	1.12	2.22	97.37	0.88	1.33	**99.91**	0.1	0.85
Phishing	96.69	0.26	4.59	93.91	0.2	2.58	94.42	0.38	1.63	**94.62**	0.3	0.99
w8a	99.11	0.04	114.85	97.5	0.07	27.22	98.12	0.12	233.78	**98.63**	0.05	63.04
a9a	84.7	0.14	373.24	78.86	0.26	74.22	82.19	0.42	39.37	**84.61**	0.18	24.27
IJCNN-1	99.27	0.7	224.29	98.41	0.05	23.56	98.07	0.09	18.7	**98.72**	0.07	14.45
Skin Segmentation	99.94	0	20.23	**99.94**	0.01	2.36	99.73	0.06	4.82	99.9	0.01	3.64
Cod-RNA	96.98	0.03	4425.55	95.44	0.03	514.9	96.19	0.03	293.29	**96.43**	0.03	152.98
$\beta = 0.3$												
Iris-Setosa	100	0	0.01	**100**	0	0.01	**100**	0	0.01	**100**	0	0.01
Iris-Versicolor	95.43	0.98	0.01	92.3	2.44	0.01	59.87	14.47	0.01	**94.01**	2.57	0.02
Iris-Virginia	96.57	2.24	0.01	93.98	2.03	0.01	**95.51**	2.53	0.01	94.31	1.77	0.02
Breast Cancer	96.99	0.89	0.01	**96.42**	0.69	0.01	96.36	0.55	0.08	96.17	0.79	0.05
Four-class	100	0	0.01	96.62	1.52	0.01	95.62	1.6	0.07	**99.5**	0.66	0.05
Checkerboard	94.3	0.63	0.03	**93.96**	1	0.01	-	-	-	87.18	2.3	0.1
German Credit	75.7	0.7	0.09	69.95	0.22	0.03	70	0.04	0.35	**70.81**	0.87	0.29
Waveform-0	89.85	0.63	1.03	89.1	0.38	0.35	88.93	0.71	0.71	**89.47**	0.34	0.41
Waveform-1	91.26	0.3	0.68	87.49	0.33	0.34	90.18	0.46	0.74	**90.8**	0.34	0.38
Waveform-2	91.76	0.48	0.94	89.63	0.43	0.33	90.71	0.66	0.84	**91.09**	0.63	0.41
Banana	90.6	0.69	0.59	81.47	0.74	0.17	86.63	1.83	0.12	**90.09**	0.49	0.1
Mushroom	100	0	1.47	97.79	0.71	2.17	97.87	0.28	1.71	**99.95**	0.07	0.97
Phishing	96.69	0.26	4.59	94.51	0.16	3	94.75	0.3	1.99	**94.93**	0.14	1.29
w8a	99.11	0.04	114.85	97.71	0.08	49.63	98.36	0.07	237.01	**98.83**	0.06	55.09
a9a	84.7	0.14	373.24	80.35	0.2	127.4	83.18	0.28	77.18	**84.68**	0.13	43.02
IJCNN-1	99.27	0.7	224.29	98.68	0.06	37.79	98.3	0.08	34.75	**98.86**	0.06	25.43
Skin Segmentation	99.94	0	20.23	**99.94**	0.01	3.84	99.75	0.08	7.01	99.91	0	3.78
Cod-RNA	96.98	0.03	4425.55	95.92	0.04	899.1	96.27	0.01	598.23	**96.48**	0.02	256.71

Table 5. Dunn's test results in different β for the significant level $\alpha = 0.05$.

β	TRSD > CVS	TRSD ≈ CVS	TRSD < CVS	TRSD > FLD	TRSD ≈ FLD	TRSD < FLD
0.1	10	6	2	9	9	0
0.2	11	4	3	9	8	1
0.3	11	4	3	10	8	0
Total	32	14	8	28	25	1

Finally, to compare the training time of SVM, CVS, FLD, and TRSD in detail, we divide it into two parts: selecting candidate vectors (SC) and training a final SVM model (TS) when $\beta = 0.3$, summarized in Table 6. From Table 6, we can notice that it especially takes longer time for TRSD and FLD than CVS to select candidate vectors with w8a dataset. This is because the time complexity of building a decision tree is $O(pn \log n)$ where p is the number of features and n is the size of training dataset. However, our proposed method takes shorter than FLD since calculating TRSD is more computationally efficient than fisher linear discriminant and is the fastest overall. The results in Table 6 show that the proposed method efficiently selects support vector candidates while maintaining good classification performance.

Table 6. Time comparison in detail. SC and TS mean computing time for selecting candidate vectors and training a final SVM model respectively.

Dataset	SVM		CVS		FLD		TRSD	
	SC	TS	SC	TS	SC	TS	SC	TS
Iris-Setosa	0	0.01	0	0.01	0	0.01	0	0.01
Iris-Versicolor	0	0.01	0	0.01	0	0.01	0	0.01
Iris-Virginia	0	0.01	0	0.01	0	0.01	0	0.01
Breast Cancer	0	0.01	0	0.01	0.07	0.01	0.04	0.01
Four-class	0	0.01	0	0.01	0.06	0.01	0.04	0.01
Checkerboard	0	0.03	0	0.01	-	-	0.09	0.01
German Credit	0	0.09	0.01	0.02	0.33	0.02	0.27	0.02
Waveform-0	0	1.03	0.18	0.17	0.58	0.13	0.3	0.11
Waveform-1	0	0.68	0.17	0.17	0.65	0.09	0.3	0.08
Waveform-2	0	0.94	0.17	0.16	0.73	0.11	0.34	0.07
Banana	0	0.59	0	0.17	0.08	0.04	0.05	0.05
Mushroom	0	1.47	1.9	0.27	0.99	0.72	0.71	0.26
Phishing	0	4.59	2.08	0.92	1.3	0.69	0.72	0.57
w8a	0	114.85	17.48	32.15	228.83	8.18	46.35	8.74
a9a	0	373.24	44.5	82.88	16.52	60.66	8.84	34.18
IJCNN-1	0	224.29	6.06	31.73	10.08	24.67	5.77	19.66
Skin Segmentation	0	20.23	0.3	3.54	3.22	3.79	2.3	1.48
Cod-RNA	0	4425.55	0.7	898.4	15.46	582.77	11.24	245.47

5. Discussion and Conclusions

In this study, we have proposed a tree-based data reduction approach for solving large-scale SVM problems. In order to reduce time consumption in training SVM models, we apply a novel support vector selection method combining tree decomposition and the proposed relative support distance. We introduce the relative distance measure along with a virtual hyperplane between two distinct adjacent regions to effectively exclude non-SV data points. The virtual hyperplane, easily obtainable, takes advantage of the decomposed tree structures and is shown to be effective in selecting support vector candidates. In computing the relative support distance, we also use the distance between each data point to the centroid in each region and combine the two in consideration of the nonlinear characteristics of support vectors. In experiments, we have demonstrated that the proposed method outperforms some existing methods for selecting support vector candidates in terms of computation time and classification performance. In the future, we would like to investigate other large-scale

SVM problems such as multi-class classification and support vector regression. We also envision an extension of the proposed method to under-sampling techniques.

Author Contributions: Investigation, M.R.; Methodology, M.R.; Software, M.R.; Writing—original draft, M.R.; Writing—review & editing, K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2020R1F1A1076278).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
2. Cristianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*; Cambridge University Press: Cambridge, UK, 2000.
3. Cai, Y.D.; Liu, X.J.; biao Xu, X.; Zhou, G.P. Support Vector Machines for predicting protein structural class. *BMC Bioinform.* **2001**, *2*, 3. [CrossRef] [PubMed]
4. Belongie, S.; Malik, J.; Puzicha, J. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 509–522. [CrossRef]
5. Ahn, H.; Lee, K.; Kim, K.j. Global Optimization of Support Vector Machines Using Genetic Algorithms for Bankruptcy Prediction. In *Proceedings of the 13th International Conference on Neural Information Processing—Volume Part III*; Springer: Berlin, Germany, 2006; pp. 420–429.
6. Bayro-Corrochano, E.J.; Arana-Daniel, N. Clifford Support Vector Machines for Classification, Regression, and Recurrence. *IEEE Trans. Neural Netw.* **2010**, *21*, 1731–1746. [CrossRef] [PubMed]
7. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. *A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory*; Association for Computing Machinery: New York, NY, USA, 1992; pp. 144–152.
8. Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Springer: Berlin, Germany, 2001; Volume 1.
9. Qiu, J.; Wu, Q.; Ding, G.; Xu, Y.; Feng, S. A survey of machine learning for big data processing. *EURASIP J. Adv. Signal. Process.* **2016**, *2016*, 1–16.
10. Liu, P.; Choo, K.K.R.; Wang, L.; Huang, F. SVM of Deep Learning? A Comparative Study on Remote Sensing Image Classification. *Soft Comput.* **2017**, *21*, 7053–7065. [CrossRef]
11. Chapelle, O. Training a support vector machine in the primal. *Neural Comput.* **2007**, *19*, 1155–1178. [CrossRef] [PubMed]
12. Platt, J.C. 12 fast training of support vector machines using sequential minimal optimization. *Adv. Kernel Methods* **1999**, 185–208.
13. Joachims, T. Svmlight: Support Vector Machine. SVM-Light Support. Vector Mach. Univ. Dortm. 1999, 19. Available online: http://svmlight.joachims.org (accessed on 29 November 2019).
14. Vishwanathan, S.; Murty, M.N. SSVM: A simple SVM algorithm. In Proceedings of the 2002 International Joint Conference on Neural Network. IJCNN'02, Honolulu, HI, USA, 12–17 May 2002.
15. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. (TIST)* **2011**, *2*, 27. [CrossRef]
16. Lee, Y.J.; Mangasarian, O.L. RSVM: Reduced Support Vector Machines. SDM. In Proceedings of the 2001 SIAM International Conference on Data Mining, Chicago, IL, USA, 5–7 April 2001; pp. 325–361.
17. Collobert, R.; Bengio, S.; Bengio, Y. A parallel mixture of SVMs for very large scale problems. *Neural Comput.* **2002**, *14*, 1105–1114. [CrossRef] [PubMed]
18. Li, M.; Chen, F.; Kou, J. Candidate vectors selection for training support vector machines. In Natural Computation, 2007. ICNC 2007. In Proceedings of the Third International Conference on Natural Computation, Haikou, China, 24–27 August 2007; pp. 538–542.
19. Nishida, K.; Kurita, T. RANSAC-SVM for large-scale datasets. In Proceedings of the 19th International Conference on Pattern Recognition, Tampa, FL, USA, 8–11 December 2008; pp. 1–4.

20. Kawulok, M.; Nalepa, J. Support Vector Machines Training Data Selection Using a Genetic Algorithm. In *Proceedings of the 2012 Joint IAPR International Conference on Structural, Syntactic, and Statistical Pattern Recognition*; Springer: Berlin, Germany, 2012; pp. 557–565.
21. Nalepa, J.; Kawulok, M. A Memetic Algorithm to Select Training Data for Support Vector Machines. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, New York, NY, USA, 12–16 July 2014; pp. 573–580.
22. Nalepa, J.; Kawulok, M. Adaptive Memetic Algorithm Enhanced with Data Geometry Analysis to Select Training Data for SVMs. *Neurocomputing* **2016**, *185*, 113–132. [CrossRef]
23. Martin, J.K.; Hirschberg, D. On the complexity of learning decision trees. International Symposium on Artificial Intelligence and Mathematics. *Citeseer* **1996**, 112–115.
24. Chang, F.; Guo, C.Y.; Lin, X.R.; Lu, C.J. Tree decomposition for large-scale SVM problems. *J. Mach. Learn. Res.* **2010**, *11*, 2935–2972.
25. Chau, A.L.; Li, X.; Yu, W. Support vector machine classification for large datasets using decision tree and fisher linear discriminant. *Future Gener. Comput. Syst.* **2014**, *36*, 57–65. [CrossRef]
26. Cervantes, J.; Garcia, F.; Chau, A.L.; Rodriguez-Mazahua, L.; Castilla, J.S.R. Data selection based on decision tree for SVM classification on large data sets. *Appl. Soft Comput.* **2015**, *37*, 787–798. [CrossRef]
27. Radial Basis Function Kernel, Wikipedia, Wikipedia Foundation. Available online: https://en.wikipedia.org/wiki/Radial_basis_function_kernel (accessed on 29 November 2019).
28. Kass, G.V. An Exploratory Technique for Investigating Large Quantities of Categorical Data. *J. R. Stat. Soc.* **1980**, *29*, 119–127. [CrossRef]
29. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; Wadsworth and Brooks: Monterey, CA, USA, 1984.
30. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993.
31. Loh, W.Y.; Shih, Y.S. Split Selection Methods for Classification Trees. *Stat. Sin.* **1997**, *7*, 815–840.
32. Martin, J.K.; Hirschberg, D.S. *The Time Complexity of Decision Tree Induction*; University of California: Irvine, CA, USA, 1995.
33. Li, X.B.; Sweigart, J.; Teng, J.; Donohue, J.; Thombs, L. A dynamic programming based pruning method for decision trees. *Inf. J. Comput.* **2001**, *13*, 332–344. [CrossRef]
34. Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **1936**, *7*, 179–188. [CrossRef]
35. Dua, D.; Graff, C. *UCI Machine Learning Repository*; University of California, School of Information and Computer Science: Irvine, CA, USA, 2019.
36. Chang, C.C.; Lin, C.J. LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1—27:27. 2011. Available online: https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/binary.html (accessed on 29 November 2019).
37. Ho, T.K.; Kleinberg, E.M. Checkerboard Data Set. 1996. Available online: https://research.cs.wisc.edu/math-prog/mpml.html (accessed on 29 November 2019).
38. Prokhorov, D. IJCNN 2001 Neural Network Competition.Slide Presentation in IJCNN'01, Ford Research Laboratory. 2001. Available online: https://www.csie.ntu.edu.tw/~{}cjlin/libsvmtools/datasets/binary.html (accessed on 29 November 2019).

applied sciences

MDPI

Article

Solving Partial Differential Equations Using Deep Learning and Physical Constraints

Yanan Guo [1,2], Xiaoqun Cao [1,2,*], Bainian Liu [1,2] and Mei Gao [1,2]

[1] College of Computer, National University of Defense Technology, Changsha 410073, China; guoyanan@nudt.edu.cn (Y.G.); bnliu@nudt.edu.cn (B.L.); gaomei17a@nudt.edu.cn (M.G.)
[2] College of Meteorology and Oceanography, National University of Defense Technology, Changsha 410073, China
* Correspondence: caoxiaoqun@nudt.edu.cn

Received: 31 July 2020; Accepted: 22 August 2020; Published: 26 August 2020

Abstract: The various studies of partial differential equations (PDEs) are hot topics of mathematical research. Among them, solving PDEs is a very important and difficult task. Since many partial differential equations do not have analytical solutions, numerical methods are widely used to solve PDEs. Although numerical methods have been widely used with good performance, researchers are still searching for new methods for solving partial differential equations. In recent years, deep learning has achieved great success in many fields, such as image classification and natural language processing. Studies have shown that deep neural networks have powerful function-fitting capabilities and have great potential in the study of partial differential equations. In this paper, we introduce an improved Physics Informed Neural Network (PINN) for solving partial differential equations. PINN takes the physical information that is contained in partial differential equations as a regularization term, which improves the performance of neural networks. In this study, we use the method to study the wave equation, the KdV–Burgers equation, and the KdV equation. The experimental results show that PINN is effective in solving partial differential equations and deserves further research.

Keywords: partial differential equations; deep learning; physics-informed neural network; wave equation; KdV-Burgers equation; KdV equation

1. Introduction

Partial differential equations (PDEs) are important tools for the study of all kinds of natural phenomena and they are widely used to explain various physical laws [1–3]. In addition, many engineering and technical problems can be modeled and analyzed using partial differential equations, such as wake turbulence, optical fiber communications, atmospheric pollutant dispersion, and so on [4–6]. Therefore, advances in partial differential equations are often of great importance to many fields, such as aerospace, numerical weather prediction, etc. [7,8]. Currently, partial differential equations and many other disciplines are increasingly connected and mutually reinforcing each other. Therefore, the study of partial differential equations is of great significance. However, a major difficulty in the study of partial differential equations is that it is often impossible to obtain analytical solutions. Therefore, various numerical methods for solving partial differential equations have been proposed by related researchers, such as the finite difference method, finite element method, finite volume method, etc. [9,10]. Numerical methods have greatly facilitated the study of partial differential equations. Nowadays, these methods have been widely used and they are being continuously improved. At the same time, researchers are also trying to develop new methods and tools to solve partial differential equations.

With the advent of big data and the enhancement of computing resources, data-driven methods have been increasingly applied [11,12]. In recent years, as a representative of data-driven methods, deep learning methods that are based on deep neural networks have made breakthrough progress [13–15]. Deep neural networks are excellent at mining various kinds of implicit information and they have achieved great success in handling various tasks in science and engineering, such as in image classification [16], natural language processing [17], and fault detection [18]. According to the universal approximation theorem, a multilayer feedforward network containing a sufficient number of hidden layer neurons can approximate any continuous function with arbitrary accuracy [19,20]. Therefore, neural networks have also tremendous advantages in function fitting. In recent years, neural network-based approaches have appeared in the study of partial differential equations. For example, Lagaris et al. [21] use artificial neural networks to solve initial and boundary value problems. They first construct a trial solution consisting of two parts, the first part satisfying the initial/boundary condition and the second part being a feedforward neural network, and then train the network to satisfy the differential equation. The experimental results show that the method has good performance. However, for high dimensional problems, the training time increases due to the larger training set, which needs to be solved by methods, such as parallel implementations. Based on the work of Lagaris et al. [21], Göküzüm et al. [22] further propose a method that is based on an artificial neural network (ANN) discretization for solving periodic boundary value problems in homogenization. In contrast to Lagaris et al. et al. [21], Göküzüm et al. [22] use a global energy potential to construct the objective to be optimized. Numerical experiments show that the method can achieve reasonable physical results using a smaller number of neurons, thus reducing the memory requirements. However, this method still faces problems, such as slow training speed and overfitting, which may be solved by dropout or regularization. Nguyen-Thanh et al. [23] propose a method to study finite deformation hyperelasticity using energy functional and deep neural networks, which is named Deep Energy Method (DEM). The method uses potential energy as a loss function and trains the deep neural network by minimizing the energy function. DEM has promising prospects for high-dimensional problems, ill-posed problems, etc. However, it also faces problems of how to add boundary conditions, integration techniques, and so on. In addition, how to better build deep neural networks is also a problem for DEM to study in depth. Although there are still many problems, deep learning methods have gradually become a new way of solving partial differential equations.

Nowadays, there has been a growing number of researchers using deep learning methods to study partial differential equations [24–27]. For example, Huang [28] combines deep neural networks and the Wiener-Hopf method to study some wave problems. This combinational research strategy has achieved excellent experimental results in solving two particular problems. It cannot be overlooked that preparing the training dataset is an important and expensive task in their study. Among the many studies, an important work that cannot be ignored is the physics-informed neural networks proposed by Raissi et al. [29–31]. This neural network model takes into account the physical laws contained in PDEs and encodes them into the neural network as regularization terms, which improves performance of the neural network model. Nowadays, physics-informed neural networks are gaining more and more attention from researchers and they are gradually being applied to various fields of research [32–36]. Jagtap et al. [37] introduce adaptive activation functions into deep and physics-informed neural networks (PINNs) to better approximate complex functions and the solutions of partial differential equations. When compared with the traditional activation functions, the adaptive activation functions have better learning ability, which improves the performance of deep and physics-informed neural networks. Based on previous studies on PINN, this study further optimizes the method and constructs physics-informed neural networks for the wave equation, KdV–Burgers equation, and the KdV equation, respectively.

The paper is structured, as follows: Section 2 introduces the proposed algorithm for solving partial differential equations based on neural networks and physical knowledge constraints. Subsequently, Section 3 provides experimental validation of the proposed method, gives the partial differential equations used and the experimental scheme, and analyzes the experimental results. In Section 4 we discuss the experimental results. Finally, Section 5 summarizes the work in this paper and lists the future work to be done.

2. Methodology

In this section, we begin with a brief introduction to neural networks. Subsequently, we present an overview of physics-informed neural networks that incorporate physical laws. The relevant algorithms and implementation framework of the PINNs are introduced.

2.1. Artificial Neural Networks

Artificial Neural Network (ANN) is a research hotspot in the field of artificial intelligence since the 1980s [38,39]. It abstracts the human brain neurons from the perspective of information processing and models various networks according to different connections. Specifically, the artificial neural network is used to simulate the process of transmitting information from neuron cells in the brain. It consists of multiple connected artificial neurons and can be used to mine and fit complex relationships hidden within the data. Besides, the connections between different neurons are given different weights, each representing the amount of influence of one neuron on another neuron. Figure 1 illustrates the structure of a feedforward neural network (FNN). Feedforward neural network [40] is a simple artificial neural network in the field of artificial intelligence. As can be seen in Figure 1, a feedforward neural network consists of an input layer, one or more hidden layers, and an output layer. Within it, parameters are propagated from the input layer through the hidden layer to the output layer. When designing a neural network, the number of hidden layers, the number of neurons per layer, and the selection of activation functions are all important factors to consider.

As the number of hidden layers increases, an artificial neuron network can be viewed as an adaptive nonlinear dynamic system that consists of a large number of neurons through various connections, which can be used to approximate a variety of complex functions. Although the structure of artificial neural networks is relatively simple, it is not easy to make artificial neural networks capable of learning. It was not until around 1980 that the backpropagation algorithm effectively solved the learning problem of multilayer neural networks, and became the most popular neural network learning algorithm [39,41]. Because an artificial neural network can be used as a function approximator, it can be considered as a learnable function and applied to solve partial differential equations. Theoretically, with enough training data and neurons, artificial neural networks can learn solutions to partial differential equations.

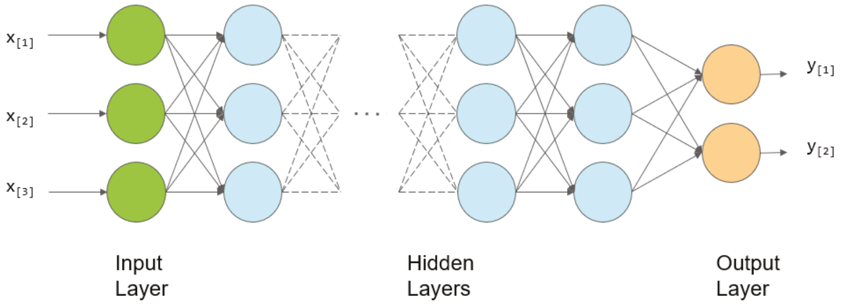


Figure 1. A structural diagram of a feedforward neural network (FNN), which consists of an input layer, one or more hidden layers, and an output layer, each containing one or more artificial neurons.

2.2. Physics-Informed Neural Networks

In this section, we introduce the physics-informed neural networks (PINNs) and related settings in this study. Traditional neural networks are based entirely on a data-driven approach that does not take into account the physical laws that are contained in the data. Therefore, a large amount of data is often required to train the neural networks to obtain a reasonable model. In contrast, physics-informed neural networks introduce physical information into the network by forcing the network output to satisfy the corresponding partial differential equations. Specifically, by adding regularization about partial differential equations to the loss function, the model is made to consider physical laws during the training process. This processing makes the training process require less data and speeds up the training process. Physics-informed neural networks can be used to solve not only the forward problem, i.e., obtaining approximate solutions to partial differential equations, but also the inverse problem, i.e., obtaining the parameters of partial differential equations from training data [29,36,42,43]. In the following, the physical-informed neural network modified and used in this study is introduced for the forward problem of partial differential equations.

In this study, consider the partial differential equation defined on the domain Ω with the boundary $\partial\Omega$.

$$\mathcal{D}(\boldsymbol{u}(\boldsymbol{x})) = \mathbf{0} \quad \boldsymbol{x} \in \Omega \tag{1}$$

$$\mathcal{B}(\boldsymbol{u}(\boldsymbol{x})) = \mathbf{0} \quad \boldsymbol{x} \in \partial\Omega \tag{2}$$

where $\boldsymbol{u}$ is the unknown solution and $\mathcal{D}$ denotes a linear or nonlinear differential operator $(e.g., \partial/\partial \boldsymbol{x}, \boldsymbol{u} \circ \partial/\partial \boldsymbol{x}, \boldsymbol{u} \circ \partial^2/\partial \boldsymbol{x}^2, etc.)$, and the operator $\mathcal{B}$ denotes the boundary condition of a partial differential Equation (e.g., Dirichlet boundary condition, Neumann boundary condition, Robin boundary condition, etc.). A point to note is that, for partial differential equations that contain temporal variables, we treat t as a special component of $\boldsymbol{x}$, i.e., the temporal domain is included in Ω. At this point, the initial condition can be treated as a special type of Dirichlet boundary condition on the spatio-temporal domain.

First, we construct a neural network for approximating the solution $u(\boldsymbol{x})$ of a partial differential equation. This neural network is denoted by $\hat{u}(\boldsymbol{x}; \boldsymbol{\theta})$, which takes the $\boldsymbol{x}$ as input and outputs a vector of the same dimension as $u(\boldsymbol{x})$. Suppose this neural network contains an input layer, $L - 1$ hidden layers and an output layer. Specifically, each hidden layer in the neural network receives the output from the previous layer and, in the k^{th} hidden layer, there are N_k number of neurons. $\boldsymbol{\theta}$ is used to represent the neural network parameters, containing the collection of weight matrix $W^{[k]} \in \mathbf{R}^{n_k \times n_{k-1}}$ and bias vector $b^{[k]} \in \mathbf{R}^{n_k}$ for each layer k with n_k neurons. These parameters will be continuously optimized during the training phase. The neural network $\hat{u}$ should satisfy two requirements: on the one hand, given a dataset of $u(\boldsymbol{x})$ observations, the network should be able to reproduce the observations when $\boldsymbol{x}$ is used as input and, on the other hand, $\hat{u}$ should conform to the physics underlying the partial differential equation. Thus, we next fulfill the requirements of the second part by defining a residual network.

$$f(\boldsymbol{x}; \boldsymbol{\theta}) := \mathcal{N}[\hat{u}(\boldsymbol{x}; \boldsymbol{\theta})] \tag{3}$$

To build this neural network, we need to use automatic differentiation (AD). Currently, automatic differentiation techniques have been widely integrated into many deep learning frameworks, such as Tensorflow [44] and PyTorch [45]. Therefore, many researchers have commonly used automatic differentiation in their studies of PINNs [29]. In this study, for the surrogate network $\hat{u}$, we derive the neural network by the automatic differentiation according to the chain rule. Moreover, since the network f has the same parameters as the network $\hat{u}$, both networks are trained by minimizing a loss function. Specifically, Figure 2 shows a schematic diagram of a physics-informed neural network.

The next main task is to find the best neural network parameters that minimize the defined loss function. In a physics-informed neural network, the loss function is defined, as follows

$$J(\boldsymbol{\theta}) = MSE_u + MSE_f \tag{4}$$

The calculation of the mean square error (MSE) is given by the following formula:

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|\hat{u}^i - u\left(x_u^i, t_u^i\right)\right|^2 \tag{5}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(x_f^i, t_f^i\right)\right|^2 \tag{6}$$

Here, $u\left(x_u^i, t_u^i\right)$ denotes training data from initial and boundary conditions and $u\left(x_f^i, t_f^i\right)$ denotes the training data in the space-time domain. Equation (5) requires the neural network to satisfy the initial and boundary conditions, while Equation (6) requires the neural network to satisfy the constraints of the partial differential equation, which corresponds to the physical information part of the neural network. Next, the optimization problem for Equation (4) is addressed by optimizing the parameters in order to find the minimum value of the loss function, i.e., we seek the following parameters.

$$w^* = \underset{w \in \theta}{\arg\min}(J(w)) \tag{7}$$

$$b^* = \underset{b \in \theta}{\arg\min}(J(b)) \tag{8}$$

In the last step, we use gradient-based optimizers to minimize the loss function, such as SGD, RMSprop, Adam, and L-BFGS [46–48]. It is found that, for smooth PDE solutions, L-BFGS can find a good solution faster than Adam, using fewer iterations. This is because Adam optimizer relies only on the first order derivative, whereas L-BFGS uses the second order derivative of the loss function [49]. However, one problem with L-BFGS is that it is more likely to get stuck on a bad local minimum. Considering their respective advantages, in this study we end up using a combination of L-BFGS and Adam optimizer to minimize the loss function. Besides, we also use a residual-based adaptive refinement (RAR) [50] method to improve the training effect by increasing the number of residual points in regions with large residuals of partial differential equations until the residuals are less than the threshold. By the above method, we will obtain trained neural networks that can be used to approximate the solutions of partial differential equations. In the next part, we will use the above method to study three important partial differential equations: the one-dimensional wave equation, the KdV–Burgers equation and the KdV equation.

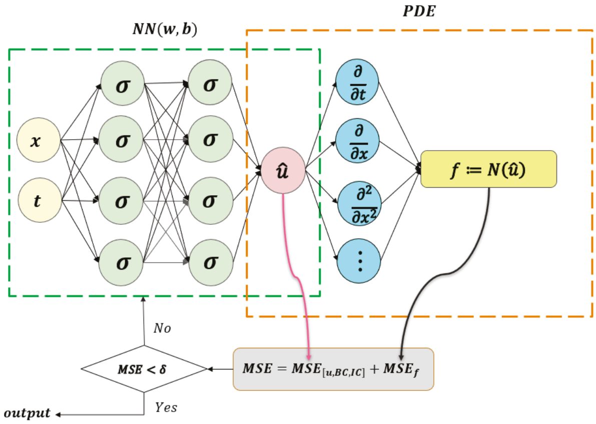


Figure 2. The schematic of physics-informed neural network (PINN) for solving partial differential equations.

3. Experiments and Results

In this section, we study the one-dimensional wave equation, the KdV-Burgers equation and the KdV equation using physics-informed neural networks. The neural network models are constructed for these three equations, respectively, based on the given initial and boundary conditions. The approximation results of the neural networks are compared with the true solutions to test the physics-informed neural networks in this paper. All of the experiments were done on Ubuntu 16.04 and we used the open-source TensorFlow to build and train the neural network models. Besides, we used PyCharm 2019.3 which is developed by JetBrains as the development environment for the experiments and NVIDIA GeForce GTX 1080 Ti. In the following, we will present the experimental design and results of these three equations, respectively.

3.1. Wave Equation

This section presents an experimental study of the wave equation using the physics-informed neural network. The wave equation is a typical hyperbolic partial differential equation and it contains second-order partial derivatives about the independent variable. In physics, the wave equation describes the path of a wave propagating through a medium and is used to study the various types of wave propagation phenomena. It appears in many fields of science, such as acoustic wave propagations, radio communications, and seismic wave propagation [51–53]. The study of wave equations is of great importance, as they are widely used in many fields. In this study, we choose a one-dimensional wave equation [54] for our experiments. In mathematical form, this wave equation is defined, as follows:

$$u_{tt} - cu_{xx} = 0, \quad x \in [0,1], \quad t \in [0,1] \tag{9}$$

where u is a function of the spatial variables x and time t. In the equation, the value of c represents the wave propagation velocity, which is given as 1 in this study. Besides, for this wave equation, its initial conditions and the homogeneous Dirichlet boundary conditions are given, as follows:

$$\begin{aligned} u(0,x) &= \tfrac{1}{2}\sin(\pi x) \\ u_t(0,x) &= \pi \sin(3\pi x) \\ u(t,0) &= u(t,1) = 0 \end{aligned} \tag{10}$$

The true solution of the above equation is $u(t,x) = \frac{1}{2}\sin(\pi x)\cos(\pi t) + \frac{1}{3}\sin(3\pi x)\sin(3\pi t)$, which we use to generate the data. The initial conditions, boundary conditions, and some random data in the space-time domain are used as training data to train the neural network model. In order to test the performance of the training model, we use the neural network model to make multiple predictions and compare it with the true solution of the partial differential equation. The specific experimental setup and procedure are as follows.

First, a neural network is designed for approximating the solutions of partial differential equations, denoted as $\hat{u}(t,x)$. For the architecture of the neural network, it contains six hidden layers, each with 100 neurons, and a hyperbolic tangent tanh is chosen as the activation function. Besides, a physics-informed neural network $f(t,x)$ is constructed for introducing control information of the equation. In our experiments, we use TensorFlow to construct the neural network. As a widely used deep learning framework, it has sophisticated automatic differentiation, so it is easy to introduce information about the equations. Specifically, the definition of a physical information neural network $f(t,x)$ is given by

$$f(t,x) := u_{tt} - u_{xx} \tag{11}$$

The next main task is to train the parameters of the neural network $\hat{u}(t,x)$ and $f(t,x)$. We continuously optimize the parameters by minimizing the mean square error loss to obtain the optimal parameters.

$$J(\boldsymbol{\theta}) = MSE_u + MSE_f \tag{12}$$

where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u\left(t_u^i, x_u^i\right) - u^i \right|^2 \tag{13}$$

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f\left(t_f^i, x_f^i\right) \right|^2 \tag{14}$$

where MSE_u is a loss function constructed using observations of initial and boundary conditions. MSE_f is a loss function that is based on partial differential equations for introducing physical information. Specifically, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ corresponds to the initial and boundary training data of $u(t,x)$, and N_u is the number of data provided. In addition, $u(t_f, x_f)$ and $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ corresponds to the training data of the spatio-temporal domain, and N_f is the corresponding number of training data. In this work, to fully consider the physical information embedded in the equations, we select the data in the spatio-temporal domain to train the neural network. The training data of the spatio-temporal domain is selected randomly, and the amount of training data N_f is 40,000. Besides, the total number of training data of the initial and boundary conditions is relatively small, and the expected effect can be achieved when N_u is 300. Similarly, the selection of training data for the initial and boundary conditions is also random. During the optimization procedure, we set the learning rate to 0.001, and in order to balance convergence speed and global convergence, we ran L-BFGS 30,000 epochs and then continued the optimization using Adam until convergence. In addition, we used the Glorot normal initializer [55] for initialization. In this experiment, the time to train the model was approximately fifteen minutes. We tested the effect of the model after completing the training of the neural network model. Figure 3 is the prediction of the neural network model obtained from the training, and it can be seen that the prediction obtained is quite complex. We choose different moments to compare the prediction with the exact solution to test the accuracy of this prediction. Figure 4 shows the comparison between the exact solution and the prediction at different times $t = 0.2, 0.5, 0.8$. From Figure 4, it can be seen that the predictions of the neural network model and exact solutions are very consistent, indicating that the constructed neural network model has a good ability to solve partial differential equations. In addition, the relative L2 error of this example was calculated to be $5.16 \cdot 10^{-4}$, which further validates the effectiveness of this method. Although the solution of the selected partial differential equations is complex, the neural network model can still approximate a result very close to the true solution from the training data, indicating that the neural network with physical information has great potential and value, and is worthy of further research.

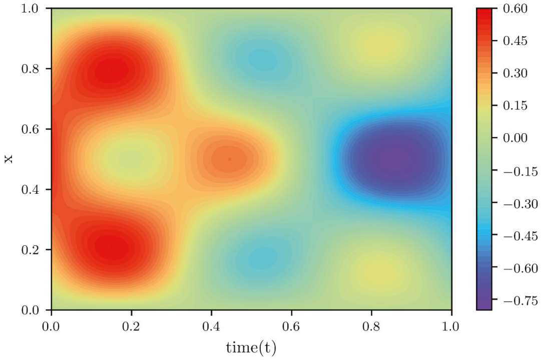


Figure 3. Solution of the wave equation given by physics-informed neural networks.

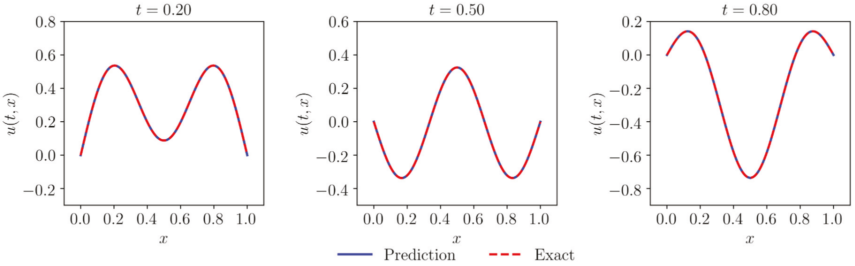


Figure 4. Comparison of the prediction given by physics-informed neural networks with the exact solution.

3.2. KdV-Burgers Equation

We have studied the KdV–Burgers equation to further analyze the ability of physics-informed neural networks to solve complex partial differential equations. The KdV–Burgers equation is a nonlinear partial differential equation containing higher-order derivatives that has been of interest to many researchers [56,57]. Today, the KdV–Burgers equation is widely studied and applied in many fields, such as the study of the flow of liquids containing bubbles, the flow of liquids in elastic tubes, and other problems [58,59]. In mathematical form, the KdV–Burgers equation is defined, as follows:

$$u_t + \alpha u u_x + \beta u_{xx} + \gamma u_{xxx} = 0 \tag{15}$$

where α, β and γ are all non-zero real constants, i.e., $\alpha\beta\gamma \neq 0$. Equation (15) can be reduced to Burgers equation [60] or Korteweg-de Vries (KdV) equation [61] in special cases. Specifically, when $\gamma = 0$, the Equation (15) is simplified to Burgers equation.

$$u_t + \alpha u u_x + \beta u_{xx} = 0 \tag{16}$$

The Burgers equation is a second-order nonlinear partial differential equation, which is used to simulate the propagation and reflection of shock waves. This equation is used in various fields of research, such as fluid dynamics and nonlinear acoustics (NLA) [60,62]. Besides, Equation (15) becomes the Korteweg-de Vries (KdV) equation when β is zero.

$$u_t + \alpha u u_x + \gamma u_{xxx} = 0 \tag{17}$$

The Korteweg-de Vries (KdV) equation was first introduced in 1985 by Korteweg and de Vries. It is a very important equation, both mathematically and practically, for the description of small amplitude shallow-water waves, ion-phonon waves, and fluctuation phenomena in biological and physical systems [63,64]. This equation, which differs from the Burgers equation in that it does not introduce dissipation and it can explain the existence of solitary waves, is of great interest to physicists and mathematicians. Therefore, the study of this equation is of great scientific significance and research value.

The KdV–Burgers equation can be viewed as a combination of the Korteweg-de Vries equation and the Burgers equation, containing the nonlinearity uu_x, the dispersion u_{xxx}, and the dissipation u_{xx}, with high complexity. The equation has been applied in many fields and it has received a great deal of attention from many researchers. In this section, we use physics-informed neural networks to develop new methods for solving the KdV–Burgers equation.

In this experiment, an important task is to construct a high-quality training data set based on partial differential equations. For Equation (15), the values of α, β, γ are given as 1,−0.075,$\pi/1000$, respectively, and deterministic initial conditions are given. In mathematical form, the nonlinear KdV–Burgers equation with periodic boundary conditions studied in this section is defined, as follows

$$\begin{aligned}
&u_t + uu_x - 0.075u_{xx} + \pi/1000u_{xxx} = 0, \quad x \in [-1,1], \quad t \in [0,1] \\
&u(0,x) = e^{(0.005*cos(\pi*x))}sin(\pi*x) \\
&u(t,-1) = u(t,1) \\
&u_x(t,-1) = u_x(t,1)
\end{aligned} \tag{18}$$

For Equation (18), we simulate it using conventional spectral methods and use the Chebfun package [65] in the programming implementation. Specifically, we integrate Equation (18) from the initial moment $t = 0$ to the final time $t = 1.0$ using a time step $t = 10^{-6}$, depending on the initial and periodic boundary conditions. Besides, we use a fourth-order explicit Runge–Kutta temporal integrator and a spectral Fourier discretization with 512 modes to ensure the accuracy of the integration.

After obtaining the high-resolution dataset, we next constructed a neural network to approximate the solution of Equation (18), which is denoted as $\hat{u}(t,x)$. The neural network contains seven hidden layers, each containing 120 neurons, and the hyperbolic tangent tanh is chosen as the activation function. Besides, the physical information neural network $f(t,x)$ is constructed using the automatic differentiation of the TensorFlow to introduce control information of the equations. Specifically, the physics-informed neural network $f(t,x)$ is defined, as follows

$$f(t,x) := u_t + uu_x - 0.075u_{xx} + \pi/1000u_{xxx} \tag{19}$$

The next main task is to train the parameters of the neural network $\hat{u}(t,x)$ and $f(t,x)$. We continuously optimize the parameters by minimizing the mean square error loss to obtain the optimal parameters.

$$J(\boldsymbol{\theta}) = MSE_u + MSE_f \tag{20}$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u\left(t_u^i, x_u^i\right) - u^i\right|^2 \tag{21}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(t_f^i, x_f^i\right)\right|^2 \tag{22}$$

Similar to the previous experiment, MSE_u is a loss function constructed while using observations of the initial and boundary conditions. MSE_f is a loss function that introduces physical information of partial differential equations. Specifically, $\left\{t_u^i, x_u^i, u^i\right\}_{i=1}^{N_u}$ corresponds to the initial and boundary condition data, $u(t_f, x_f)$ and $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ corresponds to the data in the space-time domain. In this experiment, the training data are also randomly selected from the generated dataset, with the number of training data N_f in the space-time domain being 30,000 and the number of training data N_u in the initial and boundary conditions being 400. We use the Glorot normal initializer for initialization. During optimization, we set the learning rate to 0.001 and, to ensure global convergence and speed up the convergence process, we ran L-BFGS for 30,000 epochs and then used Adam to continue optimizing until convergence. In this experiment, the time to train the model was approximately twelve minutes. After completing the training of the neural network model, we also tested the effects of the model. Figure 5 shows the predictions of the neural network model, and we can see that the resulting predictions are quite complex. We choose different moments of the prediction results to compare with the exact solution in order to test the accuracy of this prediction. Figure 6 shows the comparison between the exact solution and prediction at different moments $t = 0.25, 0.5, 0.75$. From Figure 6, it can be seen that the predictions of the neural network model and exact solutions are highly consistent, indicating that the constructed neural network model can solve the KdV–Burgers equation well. In addition, the relative L2 error of this example was calculated to be $4.79 \cdot 10^{-4}$, which further validates the effectiveness of this method. Despite the high complexity of the KdV–Burgers equation, the neural network model can still obtain results very close to the true solution from the training data, which again shows that the method has great potential and value and it is worthy of further research.

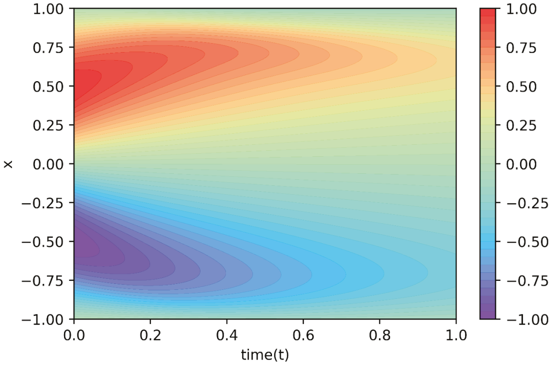


Figure 5. Solution of the Korteweg-de Vries (KdV)–Burgers equation given by physics-informed neural networks.

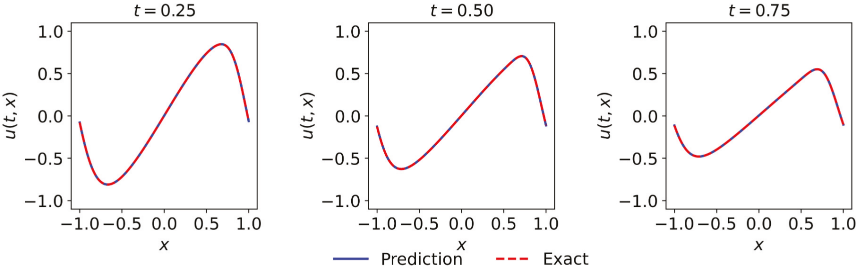


Figure 6. Comparison of the prediction given by physics-informed neural networks with the exact solution.

3.3. Two-Soliton Solution of the Korteweg-De Vries Equation

KdV equations are an important class of equations that have soliton solutions, as introduced in Section 3.2. The study of the KdV equation is important for understanding the nature of solitons and the interaction of two or more solitons. Many scholars have studied the multi-soliton solutions of KdV equations [66] and, in this section, we employ the proposed physics-informed neural networks to study the following KdV equations:

$$u_t + 6uu_x + u_{xxx} = 0, \quad -\infty < x < \infty \tag{23}$$

when given the initial condition $u(0,x) = 6\sec h^2 x$, Equation (23) has the following two-soliton solution $u(x,t)$

$$u(x,t) = 12\frac{3+4\cosh(2x-8t)+\cosh(4x-64t)}{(3\cosh(x-28t)+\cosh(3x-36t))^2} \tag{24}$$

First, because this solution is valid for either positive or negative t, we obtained data based on the true solution for $x \in [-20,20]$ and $t \in [-1,1]$. Some initial data and some random data in the space-time domain are selected as training data. Next, we constructed a neural network to approximate the solution of Equation (23), which is denoted as $\hat{u}(t,x)$. The neural network contains seven hidden layers, each containing 100 neurons, and the hyperbolic tangent tanh is chosen as the activation function. Besides, the physical information neural network $f(t,x)$ is constructed using the automatic differentiation of the TensorFlow in order to introduce control information of the equation. Specifically, the physics-informed neural network $f(t,x)$ is defined, as follows:

$$f(t,x) := u_t + 6uu_x + u_{xxx} \tag{25}$$

Next, we obtain the optimal parameters of the neural network $\hat{u}(t,x)$ and $f(t,x)$ by minimizing the following mean square error loss.

$$J(\boldsymbol{\theta}) = MSE_u + MSE_f \tag{26}$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u\left(t_u^i, x_u^i\right) - u^i\right|^2 \tag{27}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(t_f^i, x_f^i\right)\right|^2 \tag{28}$$

Similar to the previous experiment, on the one hand, the loss function MSE_u is constructed based on the observation data of the initial condition and boundary condition; on the other hand, the loss function MSE_f is constructed based on the physical information of the partial differential equation. Specifically, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ corresponds to the initial and boundary condition data, $u(t_f, x_f)$ and $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ corresponds to the data in the space-time domain. In this experiment, we randomly select training data in the data set. The number of training data in the spatio-temporal domain is 60,000, and the number of training data that meets the initial and boundary conditions is 400. We used the Glorot normal initializer for initialization and set the learning rate for the optimization process to 0.001. We ran L-BFGS for 30,000 epochs and then used Adam to continue the optimization until convergence to ensure global convergence and speed up the convergence process. In this experiment, the time to train the model was about eighteen minutes. After training the neural network model, we used the model to make predictions and compared the results with the true solution. Figure 7 shows the predictions of the KdV equation given by physics-informed neural networks. Figure 8 gives the comparison between the prediction and the exact solution at different moments $t = -0.5, 0, 0.5$ to test the performance of the model. It can be seen from Figure 8 that the prediction is very close to the exact solution. Also, it can be seen that tall wave propagates faster than short waves, which is consistent with the nature of the solitary wave solutions of the KdV equation. Thus, it can be concluded that the neural network model can simulate the KdV equation well. Besides, the relative L2 error was calculated to be $4.62 \cdot 10^{-3}$ in this case, further validating the effectiveness of the method. Because physics-informed neural networks can simulate the solitary wave solution of the KdV equation well, we will apply this method to the study of multi-soliton solutions of other equations, such as the Boussinesq equation [66,67], in future studies.

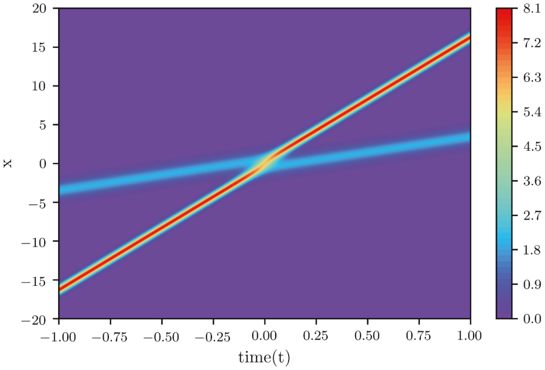


Figure 7. Two-soliton solution of the KdV equation given by physics-informed neural networks.

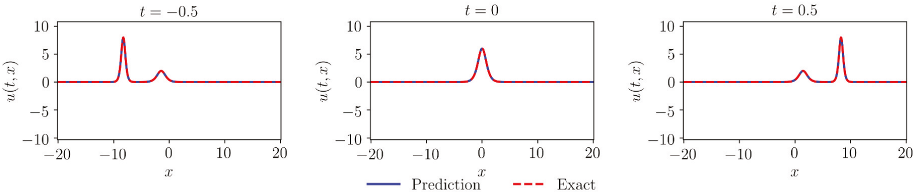


Figure 8. Comparison of the prediction given by physics-informed neural networks with the exact solution.

4. Discussions

In this study, three partial differential equations are investigated using physics-informed neural networks. Based on the characteristics of the wave equation, the KdV-Burgers equation, and the KdV equation, the corresponding neural network models were constructed and each model was experimentally analyzed. We compare the predictions of the physics-informed neural network with the true solutions of the equations and derive the following discussion.

(1) The most important task of physics-informed neural networks is to introduce a reasonable regularization of physical information. The use of physical information allows neural networks to better learn the solutions of partial differential equations from fewer observations. In this study, the physical information regularization is implemented through the automatic differentiation of the TensorFlow framework, which may not be applicable in many practical problems. Therefore, we need to develop more general differential methods and expand more methods for introducing physical information so that physics-informed neural networks can be better applied to real-world problems.

(2) Regularization has an important role in physics-informed neural networks. Similar to the previous related studies, the regularization in this study takes the form of the L2 norm. However, when considering the advantages of different norms, such as the ability of L1 norm to resist anomalous data interference, in the next study, we will adopt different forms of regularization of physical information such as L1 norm, to further improve the theory and methods related to physics-informed neural networks.

(3) In this study, the training data used to train the physics-informed neural network are randomly selected in the space-time domain, thus the physics-informed neural network does not need

to consider the discretization of partial differential equations and can learn the solutions of partial differential equations from a small amount of data. It is well known that popular computational fluid dynamics methods require consideration of the discretization of equations, such as finite difference methods. In practice, many engineering applications also need to consider the discretization of partial differential equations, for example, various numerical weather prediction models have made discretization schemes as an important part of their research. Physics-informed neural networks are well suited to solve this problem and, thus, this approach may have important implications for the development of computational fluid dynamics and even scientific computing.

(4) Although the method used in this paper has many advantages, such as not having to consider the discretization of PDEs. However, the method also faces many problems, such as the neural network for solving PDEs relies heavily on training data, which often requires more training time when the quality of the training data is poor. Therefore, it is also important to investigate how to construct high-quality training datasets to reduce the training time.

(5) In this study, we focus on solving PDEs by training physics-informed neural networks, which is a supervised learning task. Currently, several researchers have used unlabeled data to train physics-constrained deep learning models for high-dimensional problems and have quantified the uncertainty of the predictions [68]. This inspires us to further improve our neural network, so that it can be trained using unlabeled data and give a probabilistic interpretation of the prediction results [69]. Besides, this paper studies low-dimensional problems, whereas, for high-dimensional problems, model reduction [70] is also an important issue to consider when constructing a neural network model.

(6) In this paper, we study one-dimensional partial differential equations, but the method can be applied to multi-dimensional problems. Currently, we are attempting to apply the method to the simulation of three-dimensional atmospheric equations for improving existing numerical weather prediction models. Besides, in the field of engineering, complex PDE systems of field coupled nature, as in Fluid-Structure Interaction, are of great research value and they are widely used in the aerospace industry, nuclear engineering, etc. [71], so we will also study such complex PDEs in the future.

(7) So far, there have been many promising applications of neural networks in computational engineering. For example, one very interesting work is that neural networks have been used to construct constitutive laws as a surrogate model replacing the two-scale computational approaches [72,73]. These valuable works will guide us in further exploring the applications of neural networks in scientific computing.

5. Conclusions

This paper introduces a method for solving partial differential equations by neural networks that fuse physical information. The method is an improvement on previous physics-informed neural networks. In this study, the physical laws that are contained in the partial differential equations are introduced into the neural networks as a regularization. This improvement motivates the neural network to better learn the solutions of the partial differential equations from a limited number of observations. Using the method presented in this paper, we have performed the experimental analysis of three important partial differential equations. The method proposed in this paper achieves good experimental results due to the powerful function approximation ability of neural networks and the physical information contained in the partial differential equations. It is believed that, in the future, physics-informed neural networks will profoundly influence the study of solving partial differential equations and even promote the development of the whole field of scientific computing. However, there are still many problems with physics-informed neural networks, such as how to better introduce physical information into neural networks and the possible non-convergence problem in loss function optimization, which is also the next focus of our research. Besides, in the future, we will focus on analyzing the performance differences between PINN-based methods and FEM methods, comparing their accuracy, consumption time, and so on.

Author Contributions: Conceptualization, X.C. and Y.G.; methodology, Y.G. and X.C.; validation, M.G.; investigation, Y.G. and M.G.; writing—original draft preparation, Y.G.; writing—review and editing, B.L., M.G. and X.C.; visualization, Y.G. and M.G.; supervision, X.C. and B.L.; project administration, X.C. and B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (Grant No.41475094) and the National Key R&D Program of China (Grant No.2018YFC1506704).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PDEs	Partial Differential Equations
ANN	Artificial Neural Networks
FNN	Feedforward Neural Network
PINN	Physics Informed Neural Network
FEM	Finite Element Method
FDM	Finite Difference Method
FVM	Finite Volume Method
AD	Automatic Differentiation
SGD	Stochastic Gradient Descent
Adam	Adaptive Moment Estimation
L-BFGS	Limited-memory BFGS
KdV equation	Korteweg-de Vries equation

References

1. Folland, G.B. *Introduction to Partial Differential Equations*; Princeton University Press: Princeton, NJ, USA, 1995; Volume 102.
2. Petrovsky, I.G. *Lectures on Partial Differential Equations*; Courier Corporation: North Chelmsford, MA, USA, 2012.
3. Courant, R.; Hilbert, D. *Methods of Mathematical Physics: Partial Differential Equations*; John Wiley & Sons: Hoboken, NJ, USA, 2008.
4. Farlow, S.J. *Partial Differential Equations for Scientists and Engineers*; Courier Corporation: North Chelmsford, MA, USA, 1993.
5. Zauderer, E. *Partial Differential Equations of Applied Mathematics*; John Wiley & Sons: Hoboken, NJ, USA, 2011; Volume 71.
6. Churchfield, M.J.; Lee, S.; Michalakes, J.; Moriarty, P.J. A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics. *J. Turbul.* **2012**, *13*, N14. [CrossRef]
7. Müller, E.H.; Scheichl, R. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Q. J. R. Meteorol. Soc.* **2014**, *140*, 2608–2624. [CrossRef]
8. Tröltzsch, F. *Optimal Control of Partial Differential Equations: Theory, Methods, and Applications*; American Mathematical Society: Providence, RI, USA, 2010; Volume 112.
9. Ames, W.F. *Numerical Methods for Partial Differential Equations*; Academic Press: Cambridge, MA, USA, 2014.
10. Quarteroni, A.; Valli, A. *Numerical Approximation of Partial Differential Equations*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 23.
11. Yin, S.; Ding, S.X.; Xie, X.; Luo, H. A review on basic data-driven approaches for industrial process monitoring. *IEEE Trans. Ind. Electron.* **2014**, *61*, 6418–6428. [CrossRef]
12. Bai, Z.; Brunton, S.L.; Brunton, B.W.; Kutz, J.N.; Kaiser, E.; Spohn, A.; Noack, B.R. Data-driven methods in fluid dynamics: Sparse classification from experimental data. In *Whither Turbulence and Big Data in the 21st Century?* Springer: Berlin/Heidelberg, Germany, 2017; pp. 323–342.
13. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

14. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
15. Deng, L.; Yu, D. Deep learning: Methods and applications. *Found. Trends Signal Process.* **2014**, *7*, 197–387. [CrossRef]
16. Li, S.; Song, W.; Fang, L.; Chen, Y.; Ghamisi, P.; Benediktsson, J.A. Deep learning for hyperspectral image classification: An overview. *IEEE Trans. Geosci. Remote. Sens.* **2019**, *57*, 6690–6709. [CrossRef]
17. Goldberg, Y. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.* **2016**, *57*, 345–420. [CrossRef]
18. Helbing, G.; Ritter, M. Deep Learning for fault detection in wind turbines. *Renew. Sustain. Energy Rev.* **2018**, *98*, 189–198. [CrossRef]
19. Lu, Y.; Lu, J. A Universal Approximation Theorem of Deep Neural Networks for Expressing Distributions. *arXiv* **2020**, arXiv:2004.08867.
20. Hornik, K.; Stinchcombe, M.; White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks* **1990**, *3*, 551–560. [CrossRef]
21. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Networks* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
22. Göküzüm, F.S.; Nguyen, L.T.K.; Keip, M.A. An Artificial Neural Network Based Solution Scheme for Periodic Computational Homogenization of Electrostatic Problems. *Math. Comput. Appl.* **2019**, *24*, 40. [CrossRef]
23. Nguyen-Thanh, V.M.; Zhuang, X.; Rabczuk, T. A deep energy method for finite deformation hyperelasticity. *Eur. J. -Mech.-A/Solids* **2020**, *80*, 103874. [CrossRef]
24. Bar, L.; Sochen, N. Unsupervised deep learning algorithm for PDE-based forward and inverse problems. *arXiv* **2019**, arXiv:1904.05417.
25. Freund, J.B.; MacArt, J.F.; Sirignano, J. DPM: A deep learning PDE augmentation method (with application to large-eddy simulation). *arXiv* **2019**, arXiv:1911.09145.
26. Wu, K.; Xiu, D. Data-driven deep learning of partial differential equations in modal space. *J. Comput. Phys.* **2020**, *408*, 109307. [CrossRef]
27. Khoo, Y.; Lu, J.; Ying, L. Solving parametric PDE problems with artificial neural networks. *arXiv* **2017**, arXiv:1707.03351.
28. Huang, X. Deep neural networks for waves assisted by the Wiener–Hopf method. *Proc. R. Soc.* **2020**, *476*, 20190846. [CrossRef]
29. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
30. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10561.
31. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10566.
32. Chen, X.; Duan, J.; Karniadakis, G.E. Learning and meta-learning of stochastic advection-diffusion-reaction systems from sparse measurements. *arXiv* **2019**, arXiv:1910.09098.
33. He, Q.; Barajas-Solano, D.; Tartakovsky, G.; Tartakovsky, A.M. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Adv. Water Resour.* **2020**, *141*, 103610. [CrossRef]
34. Tartakovsky, A.; Marrero, C.O.; Perdikaris, P.; Tartakovsky, G.; Barajas-Solano, D. Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems. *Water Resour. Res.* **2020**, *56*, e2019WR026731. [CrossRef]
35. Kadeethum, T.; Jørgensen, T.M.; Nick, H.M. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. *PLoS ONE* **2020**, *15*, e0232683. [CrossRef] [PubMed]
36. Kissas, G.; Yang, Y.; Hwuang, E.; Witschey, W.R.; Detre, J.A.; Perdikaris, P. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **2020**, *358*, 112623. [CrossRef]
37. Jagtap, A.D.; Kawaguchi, K.; Karniadakis, G.E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.* **2020**, *404*, 109136. [CrossRef]
38. Shanmuganathan, S. Artificial neural network modelling: An introduction. In *Artificial Neural Network Modelling*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1–14.

39. Nielsen, M.A. *Neural Networks and Deep Learning*; Determination Press: San Francisco, CA, USA, 2015; Volume 2018.
40. Svozil, D.; Kvasnicka, V.; Pospichal, J. Introduction to multi-layer feed-forward neural networks. *Chemom. Intell. Lab. Syst.* **1997**, *39*, 43–62. [CrossRef]
41. Li, J.; Cheng, J.H.; Shi, J.Y.; Huang, F. Brief introduction of back propagation (BP) neural network algorithm and its improvement. In *Advances in Computer Science and Information Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 553–558.
42. Zhang, D.; Guo, L.; Karniadakis, G.E. Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks. *SIAM J. Sci. Comput.* **2020**, *42*, A639–A665. [CrossRef]
43. Tipireddy, R.; Perdikaris, P.; Stinis, P.; Tartakovsky, A. A comparative study of physics-informed neural network models for learning unknown dynamics and constitutive relations. *arXiv* **2019**, arXiv:1904.04058.
44. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
45. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. In Proceedings of the NIPS 2017 Autodiff Workshop, Long Beach, CA, USA, 9 December 2017.
46. Bengio, Y. Gradient-based optimization of hyperparameters. *Neural Comput.* **2000**, *12*, 1889–1900. [CrossRef] [PubMed]
47. Kylasa, S.; Roosta, F.; Mahoney, M.W.; Grama, A. GPU accelerated sub-sampled newton's method for convex classification problems. In Proceedings of the 2019 SIAM International Conference on Data Mining, Calgary, AB, Canada, 2–4 May 2019; pp. 702–710.
48. Richardson, A. Seismic full-waveform inversion using deep learning tools and techniques. *arXiv* **2018**, arXiv:1801.07232.
49. Le, Q.V.; Ngiam, J.; Coates, A.; Lahiri, A.; Prochnow, B.; Ng, A.Y. On optimization methods for deep learning. In Proceedings of the 28th International Conference on Machine Learning, ICML, Bellevue, WA, USA, 28 June–2 July 2011.
50. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *arXiv* **2019**, arXiv:1907.04502.
51. Li, J.; Feng, Z.; Schuster, G. Wave-equation dispersion inversion. *Geophys. J. Int.* **2017**, *208*, 1567–1578. [CrossRef]
52. Gu, J.; Zhang, Y.; Dong, H. Dynamic behaviors of interaction solutions of (3+ 1)-dimensional Shallow Water wave equation. *Comput. Math. Appl.* **2018**, *76*, 1408–1419. [CrossRef]
53. Kim, D. A Modified PML Acoustic Wave Equation. *Symmetry* **2019**, *11*, 177. [CrossRef]
54. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Numerical Gaussian processes for time-dependent and non-linear partial differential equations. *arXiv* **2017**, arXiv:1703.10230.
55. Hanin, B.; Rolnick, D. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 571–581.
56. Samokhin, A. Nonlinear waves in layered media: Solutions of the KdV–Burgers equation. *J. Geom. Phys.* **2018**, *130*, 33–39. [CrossRef]
57. Zhang, W.G.; Li, W.X.; Deng, S.E.; Li, X. Asymptotic Stability of Monotone Decreasing Kink Profile Solitary Wave Solutions for Generalized KdV-Burgers Equation. *Acta Math. Appl. Sin. Engl. Ser.* **2019**, *35*, 475–490. [CrossRef]
58. Samokhin, A. On nonlinear superposition of the KdV–Burgers shock waves and the behavior of solitons in a layered medium. *Differ. Geom. Appl.* **2017**, *54*, 91–99. [CrossRef]
59. Ahmad, H.; Seadawy, A.R.; Khan, T.A. Numerical solution of Korteweg–de Vries-Burgers equation by the modified variational iteration algorithm-II arising in shallow water waves. *Phys. Scr.* **2020**, *95*, 045210. [CrossRef]
60. Seydaoğlu, M.; Erdoğan, U.; Öziş, T. Numerical solution of Burgers' equation with high order splitting methods. *J. Comput. Appl. Math.* **2016**, *291*, 410–421. [CrossRef]
61. Khalique, C.M.; Mhlanga, I.E. Travelling waves and conservation laws of a (2+ 1)-dimensional coupling system with Korteweg-de Vries equation. *Appl. Math. Nonlinear Sci.* **2018**, *3*, 241–254. [CrossRef]

62. Wang, Y.; Navon, I.M.; Wang, X.; Cheng, Y. 2D Burgers equation with large Reynolds number using POD/DEIM and calibration. *Int. J. Numer. Methods Fluids* **2016**, *82*, 909–931. [CrossRef]
63. Wu, J.; Geng, X. Inverse scattering transform and soliton classification of the coupled modified Korteweg-de Vries equation. *Commun. Nonlinear Sci. Numer. Simul.* **2017**, *53*, 83–93. [CrossRef]
64. Khusnutdinova, K.R.; Stepanyants, Y.; Tranter, M.R. Soliton solutions to the fifth-order Korteweg–de Vries equation and their applications to surface and internal water waves. *Phys. Fluids* **2018**, *30*, 022104. [CrossRef]
65. Driscoll, T.A.; Hale, N.; Trefethen, L.N. *Chebfun Guide*; Pafnuty Publications: Oxford, UK, 2014.
66. Nguyen, L.T.K. Modified homogeneous balance method: Applications and new solutions. *Chaos Solitons Fractals* **2015**, *73*, 148–155. [CrossRef]
67. Nguyen, L.T.K. Soliton solution of good Boussinesq equation. *Vietnam. J. Math.* **2016**, *44*, 375–385. [CrossRef]
68. Zhu, Y.; Zabaras, N.; Koutsourelakis, P.S.; Perdikaris, P. Physics-Constrained Deep Learning for High-dimensional Surrogate Modeling and Uncertainty Quantification without Labeled Data. *J. Comput. Phys.* **2019**, *394*, 56–81. [CrossRef]
69. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
70. Chinesta, F.; Ladeveze, P.; Cueto, E. A Short Review on Model Order Reduction Based on Proper Generalized Decomposition. *Arch. Comput. Methods Eng.* **2011**, *18*, 395–404. [CrossRef]
71. Ohayon, R.; Schotté, J.S. Fluid–Structure Interaction Problems. In *Encyclopedia of Computational Mechanics*, 2nd ed.; American Cancer Society: Atlanta, GA, USA, 2017; pp. 1–12.
72. Nguyen-Thanh, V.M.; Nguyen, L.T.K.; Rabczuk, T.; Zhuang, X. A surrogate model for computational homogenization of elastostatics at finite strain using HDMR-based neural network. *Int. J. Numer. Methods Eng.* **2020**. [CrossRef]
73. Le, B.; Yvonnet, J.; He, Q.C. Computational homogenization of nonlinear elastic materials using neural networks. *Int. J. Numer. Methods Eng.* **2015**, *104*, 1061–1084. [CrossRef]

applied sciences

Article

Answer Set Programming for Regular Inference

Wojciech Wieczorek [1,*], Tomasz Jastrzab [2] and Olgierd Unold [3]

[1] Department of Computer Science and Automatics, University of Bielsko-Biala, Willowa 2, 43-309 Bielsko-Biala, Poland
[2] Department of Algorithmics and Software, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland; tomasz.jastrzab@polsl.pl
[3] Department of Computer Engineering, Wroclaw University of Science and Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland; olgierd.unold@pwr.edu.pl
* Correspondence: wwieczorek@ath.bielsko.pl

Received: 6 October 2020; Accepted: 25 October 2020; Published: 30 October 2020

Abstract: We propose an approach to non-deterministic finite automaton (NFA) inductive synthesis that is based on answer set programming (ASP) solvers. To that end, we explain how an NFA and its response to input samples can be encoded as rules in a logic program. We then ask an ASP solver to find an answer set for the program, which we use to extract the automaton of the required size. We conduct a series of experiments on some benchmark sets, using the implementation of our approach. The results show that our method outperforms, in terms of CPU time, a SAT approach and other exact algorithms on all benchmarks.

Keywords: answer set programming; non-deterministic automata induction; grammatical inference

1. Introduction

The main problem investigated in this paper is as follows. Given a finite alphabet Σ, two finite subsets $S_+, S_- \subseteq \Sigma^*$, and an integer $k > 0$, find a k-state NFA A that recognizes a language $L \subseteq \Sigma^*$ such that $S_+ \subseteq L$ and $S_- \subseteq \Sigma^* - L$. In other words, we are dealing with the process of learning a finite state machine based on a set of labeled strings, thus building a model reflecting the characteristics of the observations. Machine learning of automata and grammars has a wide range of applications in such fields as syntactic pattern recognition, computational biology, systems modeling, natural language acquisition, and knowledge discovery (see [1–5]).

It is well known that NFA or regular expression minimization is computationally hard: it is PSPACE-complete [6]. Moreover, even if we specify the regular language by a deterministic finite automaton (DFA), the problem remains PSPACE-complete [7]. Angluin [8] showed that there is no polynomial-time algorithm for finding the shortest compatible regular expression for arbitrary given data (if P $\neq$ NP). Thus we conjecture that the complexity of inferring a minimal-size NFA that matches a labeled set of input strings is probably exponential.

For the deterministic case, the problem is NP-complete [9]. Besides, in contrast to the NFAs, for a given regular language there is always exactly one minimum-size DFA (i.e., there is no other non-isomorphic DFA with the same minimal number of states). Therefore, is NFA induction harder than DFA induction? To answer this, let us compare the problem search space sizes expressed by the number of automata with a fixed number of states. Let c be the size of the alphabet and k the number of automaton states. The number of pairwise non-isomorphic minimal k-state DFAs over a c-letter alphabet is of order $k2^{k-1}k^{(c-1)k}$. The number of NFAs such that every state is reachable from the start state is of order 2^{ck^2} [10]. Thus, switching from determinism to non-determinism increases the search space enormously. However, on the other hand, it is well known that NFAs are more compact. A DFA could even be exponentially larger than a corresponding NFA for a given language.

The purpose of the present proposal is twofold. The first objective is to devise an algorithm for the smallest non-deterministic automaton problem. It entails preparing logical rules (this set of rules will be called an AnsProlog program) before starting the searching process. The second objective is to show how the ASP solvers help to tackle the regular inference problem for large-size instances and to compare our approach with the existing ones. Particularly, we will refer to the following exact NFA identification methods [11]:

1. A randomized algorithm using Parallelization Scheme 1 (RA-PS1), with *deg*, *mmex* and *mmcex* variable ordering methods chosen at random,
2. A randomized algorithm using Parallelization Scheme 2 (RA-PS2), with final state combinations chosen at random,
3. An ordered algorithm using Parallelization Scheme 1 (OA-PS1), with *deg*, *mmex* and *mmcex* variable ordering methods chosen according to the given order in a round robin fashion,
4. An ordered algorithm using Parallelization Scheme 2 (OA-PS2), with final state combinations ordered by the number of final states.

We will also refer to a SAT encoding given in [5]. All four above-mentioned methods and a SAT encoding are thoroughly described in Section 4.2. To enable comparisons with other methods in the future, the Python implementation of our approach is made available via GitHub. The Python scripting language is used only for generating the appropriate AnsProlog facts and running Clingo, an ASP solver.

Another line of research concerns the induction of DFAs. The original idea of SAT encoding in this context comes from the work made by Heule and Verwer [12]. Their work, in turn, was based on the idea of transformation from DFA identification into graph coloring, which was proposed by Coste and Nicolas [13]. Zakirzyanov et al. [14] proposed BFS-based symmetry breaking predicates, instead of the original max-clique predicates, which improved the translation-to-SAT technique. The improvement was demonstrated with the experiments on randomly generated input data. The core idea is as follows. Consider a graph G, the vertices of which are the states of an initial automaton and there are edges between vertices that cannot be merged. Finding minimum-size DFA is equivalent to a graph coloring with a minimum number of colors. The graph coloring constraints, on the other hand, can be efficiently encoded into SAT according to Walsh [15].

In a more recent approach, satisfiability modulo theories (SMT) are explored. Suppose that $A = (\Sigma, Q = \{0, 1, \ldots, K-1\}, s = 0, F, \delta)$ is a target automaton and P is the set of all prefixes of $S_+ \cup S_-$. An SMT encoding proposed by Smetsers et al. [16] uses four functions: $\delta\colon Q \times \Sigma \to Q$, $m\colon P \to Q$, $\lambda^A\colon Q \to \{\bot, \top\}$, $\lambda^T\colon S_+ \cup S_- \to \{\bot, \top\}$, where $\{\bot, \top\}$ represents logical $\{\text{false}, \text{true}\}$, and the following five constraints:

$$m(\varepsilon) = 0,$$

$$x \in S_+ \iff \lambda^T(x) = \top,$$

$$\forall xa \in P\colon x \in \Sigma^*,\ a \in \Sigma \quad \delta(m(x), a) = m(xa),$$

$$\forall x \in S_+ \cup S_- \quad \lambda^A(m(x)) = \lambda^T(x),$$

$$\forall q \in Q \quad \forall a \in \Sigma \quad \bigvee_{r \in Q} \delta(q, a) = r.$$

They implemented the encodings using Z3Py, the Python front-end of an efficient SMT solver Z3.

This paper is organized into five sections. In Section 2, we present necessary definitions and facts originating from automata, formal languages, and declarative problem-solving. Section 3 describes our inference algorithm based on solving an AnsProlog program. Section 4 shows the experimental results of our approach and describes in detail all reference methods. Concluding comments are made in Section 5.

2. Preliminaries

We assume the reader to be familiar with basic regular language and automata theory, for example, from [17], so that we introduce only some notations and notions used later in the paper.

2.1. Words and Languages

An *alphabet* Σ is a finite, non-empty set of symbols. A *word* w is a finite sequence of symbols chosen from an alphabet. The length of word w is denoted by $|w|$. The *empty word* ε is the word with zero length. Let x and y be words. Then xy denotes the *concatenation* of x and y, that is, the word formed by making a copy of x and following it by a copy of y. As usual, Σ^* denotes the set of words over Σ. A word w is called a *prefix* of a word u if there is a word x such that $u = wx$. It is a *proper* prefix if $x \neq \varepsilon$. A set of words taken from some Σ^*, where Σ is a particular alphabet, is called a *language*.

A *sample* S is an ordered pair $S = (S_+, S_-)$ where S_+, S_- are finite languages with an empty intersection (i.e., having no common word). S_+ is called the *positive part of* S (*examples*), and S_- the *negative part of* S (*counter-examples*).

2.2. Non-Deterministic Finite Automata

A *non-deterministic finite automaton* (NFA) is a five-tuple $A = (\Sigma, Q, s, F, \delta)$ where Σ is an alphabet, Q is a finite set of states, $s \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and δ is a relation from $Q \times \Sigma$ to Q. Members of δ are called *transitions*. A transition $((q, a), r) \in \delta$ with $q, r \in Q$ and $a \in \Sigma$, is usually written as $r \in \delta(q, a)$. Relation δ specifies the moves: the meaning of $r \in \delta(q, a)$ is that automaton A in the current state q reads a and can move to state r. If for given q and a there is no such r that $((q, a), r) \in \delta$, the automaton stops and we can assume it enters the rejecting state. Moving into a state that is not final is also regarded as rejecting but it may be just an intermediate state.

It is convenient to define $\bar{\delta}$ as a relation from $Q \times \Sigma^*$ to Q by the following recursion: $((q, ya), r) \in \bar{\delta}$ if $((q, y), p) \in \bar{\delta}$ and $((p, a), r) \in \delta$, where $a \in \Sigma$, $y \in \Sigma^*$, and requiring $((t, \varepsilon), t) \in \bar{\delta}$ for every state $t \in Q$. The *language accepted* by an automaton A is then

$$L(A) = \{x \in \Sigma^* \mid \text{there is } q \in F \text{ such that } ((s, x), q) \in \bar{\delta}\}. \tag{1}$$

Two automata are *equivalent* if they accept the same language.

Let $A = (\Sigma, Q, s, F, \delta)$ be an NFA. Then we will say that $x \in \Sigma^*$ is: (a) *recognized by accepting* (or *accepted*) if there is $q \in F$ such that $((s, x), q) \in \bar{\delta}$, (b) *recognized by rejecting* if there is $q \in Q - F$ such that $((s, x), q) \in \bar{\delta}$, and (c) *rejected* if it is not accepted.

2.3. Answer Set Programming

Let us shortly introduce the idea of answer set programming (ASP). The readers interested in the details of ASP, alternative definitions, and the formal specification of AnsProlog are referred to handbooks [18–20].

Let $\mathcal{A}$ be a set of *atoms*. A *rule* is of the form:

$$a \leftarrow b_1, \ldots, b_k, \sim c_1, \ldots, \sim c_m. \tag{2}$$

where a, b_is, and c_is are atoms and $k, m \geq 0$. The *head* of the rule, a, may be absent. The part on the right of '$\leftarrow$' is called the *body* of the rule. The symbol $\sim$ is called *default negation* and, by analogy to database systems, in logic programming it refers to the absence of information. Informally, $a \leftarrow \ldots \sim b$ means: if ... and there is no evidence for b then a should be included into a solution. A *program* Π is a finite set of rules.

Let R be the set of rules of the form:

$$a \leftarrow b_1, \ldots, b_k. \tag{3}$$

and $\mathcal{A}$ be a set of atoms occurring in R. The *model* of a set R of rules without negated atoms is a subset $M \subseteq \mathcal{A}$ which fulfills the following conditions:

1. if $a \leftarrow .$ is in R, then $a \in M$;
2. if $\leftarrow b_1, \ldots, b_k.$ is in R, then at least one b_i $(1 \leq i \leq k)$ is in $\mathcal{A} - M$; for $k = 0$ (i.e., the head and the body of a rule $r \in R$ are simultaneously absent) this condition does not hold, so no model exists;
3. for rules $a \leftarrow b_1, \ldots, b_k.$ with non-empty head and $k > 0$, $b_1, \ldots, b_k \in M$ implies $a \in M$.

Alternatively, if all atoms were treated as Boolean variables (i.e., presence is true, absence is false), M would be the model of an R exactly when all rules (i.e., clauses) are satisfied.

The semantics of a program is defined by an answer set as follows. The *reduct* Π^X of a program Π relative to a set X of atoms is defined by

$$\Pi^X = \{a \leftarrow b_1, \ldots, b_k. \mid a \leftarrow b_1, \ldots, b_k, \sim c_1, \ldots, \sim c_m. \in \Pi \text{ and } \{c_1, \ldots, c_m\} \cap X = \emptyset\}. \tag{4}$$

The $\subseteq$-smallest model of Π^X is denoted by $\mathrm{Cn}(\Pi^X)$. A set X of atoms is an *answer set* of Π if $X = \mathrm{Cn}(\Pi^X)$.

For the sake of simplicity, AnsProlog programs are written using variables (by convention, variables start with uppercase letters). Such programs are then grounded, i.e., transformed to programs with no variables, by applying a Herbrand substitution. Note, however, that clever grounding discards rules that are redundant, i.e., that can never apply, because some atoms in their bodies have no possibility to be derived [19]. For example, the program:

$el(a) \leftarrow .$
$el(b) \leftarrow .$
$equal(L, L) \leftarrow el(L).$
$neq(L, Y) \leftarrow el(L), el(Y), \sim equal(L, Y).$

can be transformed to Π:

$el(a) \leftarrow .$
$el(b) \leftarrow .$
$equal(a, a) \leftarrow el(a).$
$equal(b, b) \leftarrow el(b).$
$neq(a, a) \leftarrow el(a), el(a), \sim equal(a, a).$
$neq(a, b) \leftarrow el(a), el(b), \sim equal(a, b).$
$neq(b, a) \leftarrow el(b), el(a), \sim equal(b, a).$
$neq(b, b) \leftarrow el(b), el(b), \sim equal(b, b).$

which has a single answer set: $X = \{equal(a, a), equal(b, b), el(a), el(b), neq(b, a), neq(a, b)\}$. A reduct Π^X becomes:

$el(a) \leftarrow .$
$el(b) \leftarrow .$
$equal(a, a) \leftarrow el(a).$
$equal(b, b) \leftarrow el(b).$
$neq(a, b) \leftarrow el(a), el(b).$
$neq(b, a) \leftarrow el(b), el(a).$

Its minimal model $\mathrm{Cn}(\Pi^X)$ is just X. In other words, a set X of atoms is an answer set of a logic program Π if: (i) X is a classical model of Π and (ii) all atoms in X are justified by some rule in Π.

Recently, Answer Set Programming has emerged as a declarative problem-solving paradigm. This particular way of programming in AnsProlog is well-suited for modeling and solving problems that involve common sense reasoning. It has been fruitfully used in a range of applications.

Early ASP solvers used backtracking to find solutions. With the evolution of Boolean SAT solvers, several ASP solvers were built on top them. The approach taken by these solvers was to convert the ASP formula into SAT propositions, apply the SAT solver, and then convert the solutions back to ASP form. Newer systems, such as Clasp (which is a part of the Clingo solver, https://potassco.org/clasp/), take advantage of the conflict-driven algorithms inspired by SAT, without the complete conversion into a Boolean-logic form. These approaches improve the performance significantly, often by an order of magnitude, over earlier backtracking algorithms [21].

3. Proposed Encoding for the Induction of NFA

Our translation reduces NFA identification into an AnsProlog program. Suppose we are given a sample S over an alphabet Σ, and a positive integer k. We want to find a k-state NFA $A = (\Sigma, \{q_0, q_1, \ldots, q_{k-1}\}, q_0, F, \delta)$ such that every $w \in S_+$ is recognized by accepting and every $w \in S_-$ is recognized by rejecting. The parameter k can be regarded as the degree of data generalization. The smallest k, say k_0, for which our logic program has an answer set, will give the most general automaton. As k increases, we obtain a set of nested languages, the largest for k_0 and the smallest for some $k_m \geq k_0$. Usually, the running time for $k > k_0$ is shorter than for k_0.

Let $\mathrm{Pref}(S)$ be the set of all prefixes of $S_+ \cup S_-$. The relationship between an automaton A and a sample S in terms of ASP is constrained as shown below in seven groups of rules. In rules (5)–(24) the following convention for naming variables is used: P stands for a prefix, N stands for a number (state index), I, J, and M also represent state indexes, C stands for a character (the element of alphabet), W stands for word (which is also a prefix), U represents another prefix.

1. We have the following domain specification, i.e., our AnsProlog facts.

$$q(i) \leftarrow . \quad \text{for all } i \in \{0, 1, \ldots, k-1\}. \tag{5}$$

$$\mathit{symbol}(a) \leftarrow . \quad \text{for all } a \in \Sigma. \tag{6}$$

$$\mathit{prefix}(p) \leftarrow . \quad \text{for all } p \in \mathrm{Pref}(S). \tag{7}$$

$$\mathit{positive}(s) \leftarrow . \quad \text{for all } s \in S_+. \tag{8}$$

$$\mathit{negative}(s) \leftarrow . \quad \text{for all } s \in S_-. \tag{9}$$

$$\mathit{join}(u,\ a,\ v) \leftarrow . \quad \text{for all } u, v \in \mathrm{Pref}(S) \text{ and } a \in \Sigma \text{ such that } ua = v. \tag{10}$$

   Facts (5) and (6) define the set of states Q and the input alphabet Σ, while facts (7)–(9) describe the input sample. In particular, they define the prefixes as well as words to be recognized by accepting and rejecting, respectively.
   Finally, fact (10) defines the concatenation operation, which given prefix $u \in \mathrm{Pref}(S)$ and symbol $a \in \Sigma$ produces prefix $v \in \mathrm{Pref}(S)$.

2. The next rules ensure that in an automaton A every prefix goes to at least one state and every state is final or not.

$$x(P,\ N) \leftarrow \mathit{prefix}(P), q(N), \sim \mathit{not\_x}(P,\ N). \tag{11}$$

$$\mathit{not\_x}(P,\ N) \leftarrow \mathit{prefix}(P), q(N), \sim x(P,\ N). \tag{12}$$

$$\mathit{has\_state}(P) \leftarrow \mathit{prefix}(P), q(N), x(P,\ N). \tag{13}$$

$$\leftarrow \mathit{prefix}(P), \sim \mathit{has\_state}(P). \tag{14}$$

$$\mathit{final}(N) \leftarrow q(N), \sim \mathit{not\_final}(N). \tag{15}$$

$$\mathit{not\_final}(N) \leftarrow q(N), \sim \mathit{final}(N). \tag{16}$$

   Rules (11) and (12) describe the reachability of states $q \in Q$ by prefixes $p \in \mathrm{Pref}(S)$. State q is *reachable* by prefix p *iff* the prefix can be read by following a series of transitions from state q_0 to state q (this series of transitions builds a *path* for prefix p). The unreachable states are described by the default negation rule *not_x*. Clearly, for every prefix $p \in \mathrm{Pref}(S)$ and every state $q \in Q$, either (11) or (12) holds. Here P (a prefix) and N (a number, state index) are variables, which means that during the grounding they will be substituted for, respectively, every $p \in \mathrm{Pref}(S)$ because of the atom *prefix*(P) in the body of the rule and for every $i \in \{0, 1, \ldots, k-1\}$ because of the atom $q(N)$ in the body of the rule. Notice that for every $p \in \mathrm{Pref}(S)$ we already have fact *prefix*(p) and for every $i \in \{0, 1, \ldots, k-1\}$ we already have fact $q(i)$, which are the sources of this substitution. Rules (13) and (14) declare that for every prefix $p \in \mathrm{Pref}(S)$ there has to be some reachable state $q \in Q$. These rules follow from the fact that the members of sets S_+ and S_- have to be recognized by accepting or rejecting, respectively. In other words, for each $w \in (S_+ \cup S_-)$ there has to be at least one path in the inferred NFA.

Finally, rules (15) and (16) ensure that each state $q \in Q$ is either accepting (final) or rejecting (not final). Such rules as the pair (15) and (16) are recommended in ASP textbooks to specify that each element either is/has something or is/has not (refer for example to Chapter 4 of Chitta Baral's [18]).

3. For encoding transitions we will use predicates *delta*.

$$delta(I,\ C,\ J) \leftarrow q(I), symbol(C), q(J), \sim not\_delta(I,\ C,\ J). \tag{17}$$

$$not\_delta(I,\ C,\ J) \leftarrow q(I), symbol(C), q(J), \sim delta(I,\ C,\ J). \tag{18}$$

Rule (17) says that if there exists a transition between a pair of states $q_i, q_j \in Q$, marked with a symbol $c \in \Sigma$ then $delta(I, C, J)$ is in the model. Otherwise, the default negation rule not_delta applies (rule (18)).

4. Without sacrificing the generality, we can assume that q_0 is the initial state.

$$\leftarrow \sim x(\varepsilon,\ 0). \tag{19}$$

$$\leftarrow x(\varepsilon,\ N), q(N), N \neq 0. \tag{20}$$

Rules (19) and (20) mean that only state q_0 is reachable by the empty word ε.

5. Every counter-example has to be recognized by rejecting.

$$\leftarrow q(N), x(W,\ N), final(N), negative(W). \tag{21}$$

Recall that for the headless rules at least one predicate present in the body of the rule cannot be satisfied. Hence, rule (21) means that there is no final state that is reachable by any word $w \in S_-$.

6. Every example has to be recognized by accepting. In this rule we used an extension syntax of ASP—a choice construction. Here, it means that the number of final states, q_n, for which $((q_0, W), q_n) \in \bar{\delta}$ cannot be equal to 0 for any example w.

$$\leftarrow positive(W), \{\, final(N) : q(N), x(W,\ N) \,\} = 0. \tag{22}$$

7. Finally, there are mutual constraints between x and *delta* predicates.

$$x(W,\ M) \leftarrow q(I), q(M), join(U,\ C,\ W), x(U,\ I), delta(I,\ C,\ M). \tag{23}$$

$$\leftarrow join(U,\ C,\ W), q(N), x(W,\ N), \{\, delta(J,\ C,\ N) : q(J), x(U,\ J) \,\} = 0. \tag{24}$$

Rule (23) says that for some state r that is reachable by word $w = uc$, there exists some state q_i reachable by word u and there is a transition between states q_i and q_m with symbol c.

Similarly, rule (24) says that if there is a word $w = uc$ leading to some state $q_i \in Q$, then the number of transitions with symbol c outgoing from a state reachable by word u cannot be zero.

Example 1. *Let us see an example. Suppose we are given $S_+ = \{abc, c\}$, $S_- = \{a, ab\}$, and $k = 2$. Rules (5) to (10) concretize into:*

q(0) ←. *q(1)* ←.
symbol(a) ←. *symbol(b)* ←. *symbol(c)* ←.
prefix(ε) ←. *prefix(a)* ←. *prefix(ab)* ←. *prefix(c)* ←. *prefix(abc)* ←.
positive(c) ←. *positive(abc)* ←.
negative(a) ←. *negative(ab)* ←.
join(ε, a, a) ←. *join(ε, c, c)* ←. *join(a, b, ab)* ←. *join(ab, c, abc)* ←.

Rules (11) to (24) always remain unchanged. This program has an answer set $\{q(0), \ldots, delta(1, b, 1), final(0)\}$. In order to construct an associated NFA it is enough to take all final and delta predicates, which define, respectively, final states and transitions of the resultant automaton. So we have obtained an NFA depicted in Figure 1.

Additionally, in Appendix A there is a description of how answer sets are determined. In Appendix B a larger illustration is given.

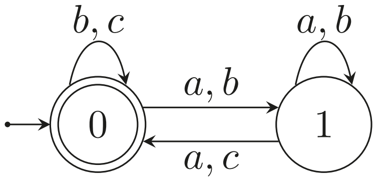


Figure 1. An inferred non-deterministic finite automaton (NFA).

4. Experimental Results

In this section, we describe some experiments comparing the performance of our approach (the program can be found at https://gitlab.com/wojtek3dan/asp4nfa) with the methods mentioned in the introductory section and described in more detail in Section 4.2. We used an ASP solver, Clingo, which can be executed sequentially or in parallel [22]. While comparing our approach with RA-PS1, RA-PS2, OA-PS1, and OA-PS2, all programs ran on an 8-core processor. ASP vs. SAT comparison was performed using a single core. For these experiments, we used a set of 40 samples (the samples can be found at https://gitlab.com/wojtek3dan/asp4nfa/-/tree/master/samples) based on randomly generated regular expressions.

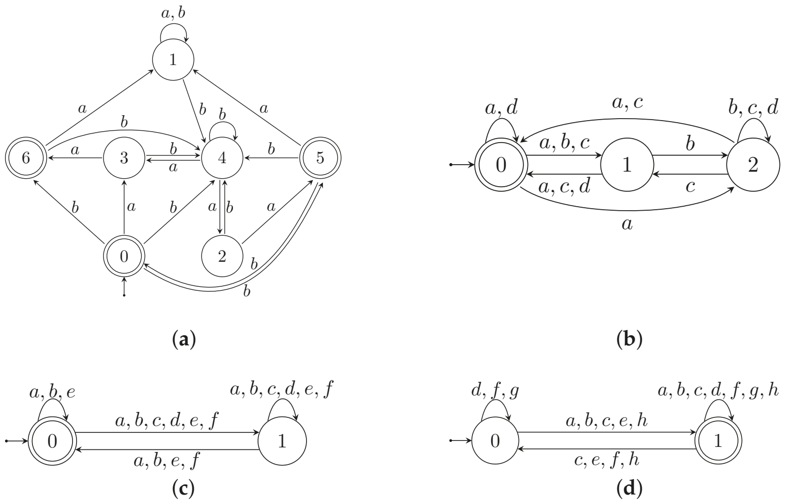
4.1. Benchmarks

As far as we know, all standard benchmarks are too hard to be solved by pure exact algorithms. Thus, we generated problem instances using our own algorithm. This algorithm builds a set of words with the following parameters: size $|E|$ of a regular expression to be generated, alphabet size $|\Sigma|$, the number $|S|$ of words actually generated and their minimum, $d_{\min}$, and maximum, $d_{\max}$, lengths. The algorithm is arranged as follows. First, construct a random regular expression E. Next, obtain corresponding minimum-state DFA M. Then, as long as a sample S is not symmetrically structurally complete (refer to Chapter 6 of [3] for the formal definition of this concept) with respect to M, repeat the following steps: (a) using the Xeger library (https://pypi.org/project/xeger/) for generating random strings from a regular expression, get two words u and w; (b) truncate as few symbols from the end of w as possible in order to obtain a counter-example $\bar{w}$; if it succeeds, add u to S_+ and $\bar{w}$ to S_-. Finally, accept $S = (S_+, S_-)$ as a valid sample if it is not too small, too large or highly imbalanced. In order to ensure that these conditions are fulfilled, the equations $|S_+| \geq 8$, $|S_-| \geq 8$, and $|S_+| + |S_-| \leq 1000$ hold for all our samples. In generating a random word from a regex or from an automaton we encounter a problem with, respectively, star operator and self-loops. Theoretically, there are infinitely many words matched to these fragments, so we have to bound the number of repetitions. We set this parameter to four.

In this manner we produced 40 sets with: $|E| \in [27, 46]$, $|\Sigma| \in \{2, 4, 6, 8\}$, $|S| \in [27, 958]$, $d_{\min} = 0$, and $d_{\max} = 305$. The file names with samples have the form 'a$|\Sigma|$words$|E|$.txt'. To give the reader a hint on the variability of the resulting automata, we show in Table 1 the numbers of states and transitions in each of the 40 NFAs found using our approach. We show there also the size of M and the size of minimal DFA D compatible with sample data. Example solutions from each group of problems, defined by the size of the input alphabet $|\Sigma|$ are also shown in Figure 2.

Table 1. Sizes of NFAs found by the answer set programming (ASP) solver (k_0—number of states, t—number of transitions, $|M|$—number of states in deterministic finite automaton (DFA) M, $|D|$—number of states in minimal DFA D compatible with sample data).

Problem	k_0	t	$\|M\|$	$\|D\|$	Problem	k_0	t	$\|M\|$	$\|D\|$
a2words27	8	32	20	15	a6words27	3	23	9	3
a2words28	3	6	7	3	a6words28	4	50	15	4
a2words29	5	22	10	6	a6words29	3	24	6	3
a2words30	3	7	6	3	a6words30	2	14	8	2
a2words31	6	13	12	7	a6words31	2	19	11	2
a2words32	5	10	10	5	a6words32	2	11	6	2
a2words33	7	19	17	8	a6words33	7	97	17	10
a2words34	4	13	6	5	a6words34	5	58	16	6
a2words35	3	6	6	3	a6words35	2	14	6	2
a2words36	5	23	9	8	a6words36	4	41	15	5
a4words27	4	32	10	4	a8words37	2	19	8	3
a4words28	3	16	14	3	a8words38	3	34	6	3
a4words29	5	34	6	6	a8words39	2	27	11	2
a4words30	3	15	7	4	a8words40	3	30	13	3
a4words31	4	23	12	4	a8words41	5	79	11	6
a4words32	3	18	6	3	a8words42	4	65	21	5
a4words33	4	24	12	4	a8words43	4	60	15	4
a4words34	4	21	16	4	a8words44	2	23	7	2
a4words35	4	28	23	4	a8words45	5	66	15	7
a4words36	2	9	5	3	a8words46	3	27	9	3



(**a**) (**b**)

(**c**) (**d**)

Figure 2. Example solutions found by the ASP solver for problems a2words33, a4words28, a6words31, and a8words37. (**a**) Problem a2words33; (**b**) Problem a4words28; (**c**) Problem a6words31; (**d**) Problem a8words37.

4.2. Compared Algorithms

As already mentioned, our algorithm was compared with a SAT-based algorithm and several exact parallel algorithms. To make the paper self-contained let us briefly describe these algorithms.

The SAT-based algorithm defines three types of binary variables, x_{wq}, y_{apq}, and z_q, for $w \in \text{Pref}(S)$, $a \in \Sigma$, $p, q \in Q$. Variable $x_{wq} = 1$ *iff* state q is reachable by prefix w, otherwise $x_{wq} = 0$. Variable $y_{apq} = 1$ *iff* there exists a transition from state p to state q with symbol a, otherwise $y_{apq} = 0$. Finally, $z_q = 1$ *iff* state q is final, and $z_q = 0$ otherwise. The constraints involving these variables are as follows:

1. All examples have to be accepted, while none of the counter-examples should be, which is described by

$$\forall_{w \in S_+ - \{\varepsilon\}} \sum_{q \in Q} x_{wq} z_q \geq 1, \tag{25}$$

$$\forall_{w \in S_- - \{\varepsilon\}} \sum_{q \in Q} x_{wq} z_q = 0. \tag{26}$$

2. All prefixes $w = a$, $w \in \text{Pref}(S), a \in \Sigma$, result from the transitions outgoing from state q_0

$$\forall_{w=a} x_{wq} - y_{aq_0q} = 0. \tag{27}$$

3. For all states $q \in Q$ reachable by prefixes $w = va$, $v, w \in \text{Pref}(S), a \in \Sigma$, there has to be some state r reachable by prefix v, and there has to be an outgoing transition from r to q with symbol a. By symmetry, if there exists a path for prefix v ending in some state r and there exists a transition from r to q with symbol a then there exists a path to state q with prefix $w = va$. These conditions are expressed as

$$\forall_{w=va} - x_{wq} + \sum_{r \in Q} x_{vr} y_{arq} \geq 0, \tag{28}$$

$$\forall_{q,r \in Q} x_{wq} - x_{vr} y_{arq} \geq 0. \tag{29}$$

Additionally, it holds that $z_{q_0} = 1$, when $\varepsilon \in S_+$, $z_{q_0} = 0$, when $\varepsilon \in S_-$, and z_{q_0} is not predefined when $\varepsilon \notin (S_+ \cup S_-)$. The solution to the presented problem formulation is sought by a SAT solver.

Example 2. *Let us consider Example 1 again. In the SAT-based formulation we have the following variables* x_{aq_0}, x_{aq_1}, …, x_{abcq_1}, $y_{aq_0q_0}$, $y_{aq_0q_1}$, …, $y_{cq_1q_1}$, z_{q_0}, *and* z_{q_1}. *Constraints (25)–(29) remain unchanged. A set of assignments satisfying the constraints at hand is as follows:* $x_{aq_1} = 1$, $x_{abq_1} = 1$, $x_{cq_0} = 1$, $x_{abcq_0} = 1$, $y_{aq_0q_1} = 1$, $y_{bq_1q_1} = 1$, $y_{cq_0q_0} = 1$, $y_{cq_1q_0} = 1$, $z_{q_0} = 1$. *All remaining variables are zeros. The resulting NFA is shown in Figure 3. Note that even though the set of transitions in Figure 3 is smaller than in Figure 1 both solutions are valid.*

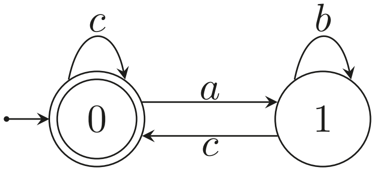


Figure 3. An inferred NFA.

Identification of a k-state NFA by means of the exact algorithms RA-PS1, RA-PS2, OA-PS1, and OA-PS2 is based on the SAT formulation given before. Assuming k is fixed we only need to determine the set of final states F and the transition function δ. Let us recall that a set of final states is *feasible iff* the following conditions are satisfied: (i) $F \neq \emptyset$, (ii) $q_0 \in F$, if $\varepsilon \in S_+$, (iii) $q_0 \notin F$, if $\varepsilon \in S_-$. Clearly, an NFA without final states cannot accept any word, and if the empty word ε is in S_+ (resp. S_-) the initial state q_0 has to be final (resp. not final). Since every feasible set F may lead to an NFA consistent with the sample S (as the NFAs need not be unique), we distribute the different sets F among processes and try to identify the δ function by means of a backtracking algorithm.

While searching for the values of y_{apq} variables, we apply different search orders. This is so, because there is no universal ordering method assuring fast convergence to the solution. The orderings used in the analyzed algorithms are *deg*, *mmex*, and *mmcex*. The *deg* ordering is a static ordering method based on a variable degree, i.e., the number of constraints the variables are involved in. The ordering does not change as the algorithm progresses. The *mmex* and *mmcex* orderings change dynamically while the algorithm runs. They aim at satisfying first the equations related to examples, or counter-examples, respectively.

The Parallelization Scheme 1 (PS1) maximizes the number of sets F processed simultaneously. If the number of available processes is greater than the number of sets F to be analyzed, we assign multiple variable orderings (VOs) to each set. In the RA-PS1 algorithm this assignment is performed randomly, while in the OA-PS1 algorithm, the *deg*, *mmex*, and *mmcex* methods are ordered by their complexity and chosen in a round robin fashion.

The Parallelization Scheme 2 (PS2) maximizes the number of variable orderings applied to the same set F. This way we shorten the time needed to obtain an answer whether an NFA exists for the given set F. If the number of available processes is smaller than the product of the number of sets F and the number of variable orderings used, we need to choose the sets F to be processed first. In the RA-PS2 algorithm we choose them at random, while in the OA-RS2 algorithm, we analyze first the sets for which the size of F is smaller.

Example 3. *Let us consider the problem given in Example 1. Since* $k = 2$ *and* $\varepsilon \notin (S_+ \cup S_-)$, *the following sets F can be defined:* $F_1 = \{q_0\}$, $F_2 = \{q_1\}$, *and* $F_3 = \{q_0, q_1\}$. *Let us also assume that we can use the three VOs discussed before. Finally, let the number of processes* $p = 3$ *(denoted by* p_i*, for* $i = 0, 1, 2$*). We can have the following example configurations of algorithms RA-PS1, RA-PS2, OA-PS1, OA-PS2:*

1. *Algorithm RA-PS1—process* p_0 *gets* (F_1, VO_3)*; process* p_1 *gets* (F_2, VO_2)*; process* p_2 *gets* (F_3, VO_3)*. Each process uses a single VO to analyze one of the possible sets* F_i, $i = 1, 2, 3$*. There is no guarantee that all VOs are used at least once.*

2. *Algorithm RA-PS2—process p_0 gets (F_2, VO_1); process p_1 gets (F_2, VO_2); process p_2 gets (F_2, VO_3). Each process uses a different VO to analyze just one set F at a time (chosen randomly). If there is no solution for set F_2, we need to repeat the above assignments but this time for the set F_1 or F_3 (again chosen randomly). We repeat the above procedure until the solution is found.*
3. *Algorithm OA-PS1—process p_0 gets (F_1, VO_1); process p_1 gets (F_2, VO_2); process p_2 gets (F_3, VO_3). Each process uses a single VO to analyze one of the possible sets F_i, $i = 1, 2, 3$, but this time the VOs are assigned according to a predefined order.*
4. *Algorithm OA-PS2—process p_0 gets (F_1, VO_1); process p_1 gets (F_1, VO_2); process p_2 gets (F_1, VO_3). Each process uses a different VO to analyze just one set F at a time, but this time we start with F_1 followed by F_2 and F_3 (unless the solution is found at some stage).*

Note that in Parallelization Scheme 2, obtaining a negative answer, i.e., that an NFA does not exist for the given set F_i, by means of one VO allows us to stop the execution of other VOs and move on to another set F_j, $i \neq j$.

4.3. Performance Comparison

In all experiments, we used Intel (Santa Clara, California, U.S.) Xeon CPU E5-2650 v2, 2.6 GHz (8 cores, 16 threads), under Ubuntu 18.04 operating system with 190 GB RAM. The time limit (TL) was set to 1000 s. The results are listed in Table 2. In order to determine whether the observed mean difference between ASP and the remaining methods is a real CPU time decrease, we used a paired samples *t* test [23] pp. 1560–1565, for ASP vs. SAT, ASP vs. RA-PS1, ASP vs. RA-PS2, ASP vs. OA-PS1, and ASP vs. OA-PS2. As we can see from Table 3, *p* value is low in all cases, so we can conclude that our results did not occur by chance and that using our ASP encoding is likely to improve CPU time performance for prepared benchmarks.

Let us explain how the mean values were computed. All TL cells were substituted by 1000. Notice that this procedure does not violate the significance of the statistical tests, because our program completed computations within the time limit for all problems (files). Thus, determining all running times would even strengthen our hypothesis.

To make the advantage of the ASP-based approach over the exact parallel algorithms even more convincing let us analyze the largest sizes of automata analyzed by the algorithms within the time limit TL = 1000 s. The summary of obtained sizes is given in Table 4. Note that the table includes only the problems for which TL entries exist in Table 2. The entries marked with * denote executions in which the algorithms started running for the given *k* but were terminated due to the time limit, without producing the final NFA.

Table 2. Execution times of exact solving NFA identification in seconds.

Problem	Execution						
	Sequential		**Parallel**				
	SAT	**ASP**	**RA-PS1**	**RA-PS2**	**OA-PS1**	**OA-PS2**	**ASP**
a2words27	148.11	77.46	TL	TL	TL	TL	10.69
a2words28	1.86	0.18	1.07	1.15	1.28	1.19	0.25
a2words29	6.40	0.49	TL	TL	TL	TL	0.40
a2words30	2.78	0.39	9.14	8.52	9.55	9.08	0.25
a2words31	2.08	0.52	TL	TL	TL	TL	0.37
a2words32	791.19	4.50	TL	TL	TL	TL	3.50
a2words33	49.72	1.99	TL	TL	TL	TL	0.93
a2words34	10.05	0.57	20.59	20.92	19.88	19.02	0.59
a2words35	0.47	0.09	0.53	0.38	0.38	0.38	0.10
a2words36	526.07	2.57	TL	TL	TL	TL	3.47

Table 2. *Cont.*

Problem	Execution						
	Sequential		Parallel				
	SAT	ASP	RA-PS1	RA-PS2	OA-PS1	OA-PS2	ASP
a4words27	1.32	0.15	TL	TL	TL	TL	0.17
a4words28	0.13	0.04	6.45	6.89	7.23	4.49	0.05
a4words29	78.93	0.96	TL	TL	TL	TL	1.15
a4words30	0.20	0.05	0.48	0.50	0.48	0.44	0.06
a4words31	21.78	0.70	TL	TL	TL	TL	1.08
a4words32	0.83	0.13	1.39	2.10	1.24	1.23	0.13
a4words33	7.19	0.48	729.40	719.59	738.89	733.21	0.50
a4words34	20.33	0.81	TL	TL	TL	TL	0.82
a4words35	21.00	0.65	TL	TL	TL	TL	0.71
a4words36	0.77	0.16	0.76	0.86	0.93	0.90	0.13
a6words27	1.13	0.15	1.59	2.24	2.89	1.88	0.16
a6words28	4.62	0.37	TL	TL	TL	TL	0.33
a6words29	1.05	0.13	3.15	2.38	2.48	2.63	0.16
a6words30	1.97	0.20	0.85	0.52	1.11	1.06	0.21
a6words31	1.43	0.18	0.86	0.81	0.94	0.83	0.22
a6words32	1.15	0.16	2.10	1.97	2.13	2.16	0.16
a6words33	441.95	4.14	TL	TL	TL	TL	2.03
a6words34	272.27	1.88	TL	TL	TL	TL	1.86
a6words35	0.20	0.06	0.85	0.73	0.82	0.82	0.06
a6words36	15.97	0.86	TL	TL	TL	TL	0.81
a8words37	0.20	0.06	0.46	0.50	0.37	0.41	0.06
a8words38	63.71	1.40	15.73	16.15	16.09	18.08	1.24
a8words39	1.44	0.19	1.55	1.55	1.78	1.48	0.20
a8words40	7.63	0.44	TL	TL	TL	TL	0.66
a8words41	324.29	1.94	TL	TL	TL	TL	2.61
a8words42	5.86	0.35	TL	TL	TL	TL	0.42
a8words43	9.14	0.45	TL	TL	TL	TL	0.69
a8words44	1.23	0.21	1.52	1.34	1.25	1.22	0.38
a8words45	279.74	2.07	TL	TL	TL	TL	2.32
a8words46	5.20	0.33	TL	TL	TL	TL	0.58
Mean	78.29	2.71	544.96	544.73	545.24	545.01	1.01

Table 3. Obtained p values from the paired samples t test.

ASP vs. SAT	ASP vs. RA-PS1	ASP vs. RA-PS2	ASP vs. OA-PS1	ASP vs. OA-PS2
0.00773	2.72×10^{-8}	2.73×10^{-8}	2.70×10^{-8}	2.72×10^{-8}

Table 4. Sizes of NFAs reached by the parallel algorithms. The sign * means that the time limit was exceeded.

Problem	ASP	RA-PS1	RA-PS2	OA-PS1	OA-PS2
a2words27	8	6	6	6	6
a2words29	5	5 *	5 *	5 *	5 *
a2words31	6	5	5	5	5
a2words32	5	4	4	4	4
a2words33	7	5	5	5	5
a2words36	5	4	4	4	4
a4words27	4	4 *	4 *	4 *	4 *
a4words29	5	4	4	4	4
a4words31	4	4 *	4 *	4 *	4 *
a4words34	4	3	3	3	3
a4words35	4	4 *	4 *	4 *	4 *

Table 4. *Cont.*

Problem	ASP	RA-PS1	RA-PS2	OA-PS1	OA-PS2
a6words28	4	3	3	3	3
a6words33	7	3	3	3	3
a6words34	5	3	3	3	3
a6words36	4	3	3	3	3
a8words40	3	3 *	3 *	3 *	3 *
a8words41	5	3	3	3	3
a8words42	4	4 *	4 *	4 *	4 *
a8words43	4	3	3	3	3
a8words45	5	4	4	4	4
a8words46	3	3 *	3 *	3 *	3 *

5. Conclusions

We have experimented with a model learning approach based on ASP solvers. The approach is very flexible, as proven by its successful adaptation for learning NFAs, implemented in the provided open source tool. Experiments indicate that our approach clearly outperforms the current state-of-the-art satisfiability-based method and all backtracking algorithms proposed in the literature. The approach does scale well (as far as non-deterministic acceptors are considered): we have shown that it can be used for learning models from up to a thousand words. In the future, we wish to develop more efficient encodings that will make the approach scale even better. We hope this paper encourages more interest in ASP-based problem solving since the presented approach has several benefits over traditional model learning algorithms. The ASP encoding is more readable than SAT encoding and the resulting program is much faster than its backtracking counterparts.

Author Contributions: Conceptualization, W.W.; methodology, O.U. and W.W.; software, T.J. and W.W.; validation, T.J. and O.U.; formal analysis, O.U.; investigation, W.W. and T.J.; resources, W.W.; writing—original draft preparation, W.W. and T.J.; writing—review and editing, O.U. and T.J.; supervision, O.U.; project administration, O.U.; funding acquisition, O.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Science Center (Poland), grant number 2016/21/B/ST6/02158.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. How Answer Sets Are Computed

Let Π be a grounded program, and let $\mathcal{A}$ be a set of atoms occurring in Π. Assume that atom z is not in $\mathcal{A}$. Observe that every grounded program Π that has rules

$$\leftarrow b_1, \ldots, b_k, \sim c_1, \ldots, \sim c_m. \tag{A1}$$

with empty head, can be transformed to a program without such rules by inserting z and $\sim z$ in this manner:

$$z \leftarrow b_1, \ldots, b_k, \sim c_1, \ldots, \sim c_m, \sim z. \tag{A2}$$

A grounded program without empty-headed rules will be called *normal*.

A rule r of the form:

$$a \leftarrow b_1, \ldots, b_k. \tag{A3}$$

where there is no default negation in the body, and the head is not empty, will be called *positive*. A program that contains only positive rules will be called positive too. We will denote by $\text{head}(r)$ the set $\{a\}$, and by $\mathrm{body}(r)$ the set $\{b_1, \ldots, b_k\}$. Now, let us define how a positive program P can act on a set of atoms $X \subseteq \mathcal{A}$ (here $\mathcal{A}$ is a set of atoms occurring in P):

$$X^P = \{\mathrm{head}(r) \mid r \in P \text{ and } \mathrm{body}(r) \subseteq X\}. \tag{A4}$$

This operation can be repeated and we define:

$$X^{P1} = X^P \quad \text{and} \quad X^{Pi} = (X^{Pi-1})^P. \tag{A5}$$

It is easy to see that $\mathrm{Cn}(P) = \bigcup_{i \geq 1} \emptyset^{Pi}$. Because for a certain i the equation $X^{Pi} = X^{Pi+1}$ holds, determining $\mathrm{Cn}(P)$ is straightforward and fast.

Consider any normal program Π. We recall from Section 2.3 that a set $X \subseteq \mathcal{A}$ is an answer set of Π if $X = \mathrm{Cn}(\Pi^X)$ (please do not confuse the reduct with program's acting on a set of atoms). Take two sets, L and U, such that $L \subseteq X \subseteq U$ for an answer set X of Π. Observe that: (i) $X \subseteq \mathrm{Cn}(\Pi^L)$, and (ii) $\mathrm{Cn}(\Pi^U) \subseteq X$. Thus we get:

$$L \cup \mathrm{Cn}(\Pi^U) \subseteq X \subseteq U \cap \mathrm{Cn}(\Pi^L). \tag{A6}$$

The last property is a recipe for expanding the lower bound L and cutting down the upper bound U. The procedure in which we replace L by $L \cup \mathrm{Cn}(\Pi^U)$ and then U by $U \cap \mathrm{Cn}(\Pi^L)$ as long as L or U are changed, will be called *narrowing*. At some point we get $L = U = X$. When we start from $L = \emptyset$, $U = \mathcal{A}$, then there are also two more possibilities: $L \not\subseteq U$ (there is no answer set), and $L \subset U$. In the latter case we can take any $a \in U - L$ and check out two paths: a should be included into L or a should be excluded from U. This leads to Algorithm A1 [19]:

Algorithm A1: Final algorithm

```
Solve($\Pi, L, U$)
    $(L, U) \leftarrow$ narrowing($\Pi, L, U$)
    if $L \not\subseteq U$ then return
    if $L = U$ then output $L$
    else
        choose $a \in U - L$
        Solve($\Pi$, $L \cup \{a\}$, $U$)
        Solve($\Pi$, $L$, $U - \{a\}$)
```

Which outputs all answer sets of a program Π provided that it had been invoked with Solve(Π, $\emptyset$, $\mathcal{A}$). The pessimistic time complexity of this algorithm can be assessed by the recurrence relation $T(n) = 2T(n-1) + n^2$, where $n = |U| - |L|$, which gives us the exponential complexity $T(n) = O(2^n)$.

Appendix B. The Complete Example of an ASP Program for NFA Induction

Suppose we are given $\Sigma = \{a, b\}$, $S_+ = \{a\}$, $S_- = \{b\}$, and $k = 2$. After grounding rules (5)–(24) we get a program Π in a Clasp format (symbol `:-` denotes left arrow, symbol `not` denotes default negation, and symbol `lambda` denotes the empty word ε):

```
symbol(a).
symbol(b).
prefix(lambda).
prefix(a).
prefix(b).
positive(a).
negative(b).
join(lambda,a,a).
join(lambda,b,b).
q(0).
q(1).
delta(0,a,0):-not not_delta(0,a,0).
delta(1,a,0):-not not_delta(1,a,0).
delta(0,b,0):-not not_delta(0,b,0).
delta(1,b,0):-not not_delta(1,b,0).
delta(0,a,1):-not not_delta(0,a,1).
delta(1,a,1):-not not_delta(1,a,1).
delta(0,b,1):-not not_delta(0,b,1).
delta(1,b,1):-not not_delta(1,b,1).
not_delta(0,a,0):-not delta(0,a,0).
not_delta(1,a,0):-not delta(1,a,0).
not_delta(0,b,0):-not delta(0,b,0).
not_delta(1,b,0):-not delta(1,b,0).
not_delta(0,a,1):-not delta(0,a,1).
not_delta(1,a,1):-not delta(1,a,1).
not_delta(0,b,1):-not delta(0,b,1).
not_delta(1,b,1):-not delta(1,b,1).
x(lambda,0):-not not_x(lambda,0).
x(a,0):-not not_x(a,0).
x(b,0):-not not_x(b,0).
x(lambda,1):-not not_x(lambda,1).
x(a,1):-not not_x(a,1).
x(b,1):-not not_x(b,1).
not_x(lambda,0):-not x(lambda,0).
not_x(a,0):-not x(a,0).
not_x(b,0):-not x(b,0).
not_x(lambda,1):-not x(lambda,1).
not_x(a,1):-not x(a,1).
not_x(b,1):-not x(b,1).
x(a,0):-delta(1,a,0),x(lambda,1).
x(a,1):-delta(1,a,1),x(lambda,1).
x(b,0):-delta(1,b,0),x(lambda,1).
x(b,1):-delta(1,b,1),x(lambda,1).
x(a,0):-delta(0,a,0),x(lambda,0).
x(a,1):-delta(0,a,1),x(lambda,0).
x(b,0):-delta(0,b,0),x(lambda,0).
x(b,1):-delta(0,b,1),x(lambda,0).
:-x(a,0),0>=#count{0,delta(0,a,0):x(lambda,0),delta(0,a,0);
0,delta(1,a,0):delta(1,a,0),x(lambda,1)}.
:-x(a,1),0>=#count{0,delta(0,a,1):x(lambda,0),delta(0,a,1);
0,delta(1,a,1):x(lambda,1),delta(1,a,1)}.
:-x(b,0),0>=#count{0,delta(0,b,0):x(lambda,0),delta(0,b,0);
0,delta(1,b,0):x(lambda,1),delta(1,b,0)}.
:-x(b,1),0>=#count{0,delta(0,b,1):x(lambda,0),delta(0,b,1);
0,delta(1,b,1):x(lambda,1),delta(1,b,1)}.
final(0):-not not_final(0).
final(1):-not not_final(1).
not_final(0):-not final(0).
not_final(1):-not final(1).
:-0>=#count{0,final(0):final(0),x(a,0);0,final(1):final(1),x(a,1)}.
:-final(0),x(b,0).
:-final(1),x(b,1).
:-x(lambda,1).
```

```
:-not x(lambda,0).
has_state(lambda):-x(lambda,0).
has_state(a):-x(a,0).
has_state(b):-x(b,0).
has_state(lambda):-x(lambda,1).
has_state(a):-x(a,1).
has_state(b):-x(b,1).
:-not has_state(lambda).
:-not has_state(a).
:-not has_state(b).
```

One of the answer sets X is:

```
q(0) q(1) prefix(lambda) prefix(a) prefix(b) symbol(a) symbol(b)
negative(b) positive(a) join(lambda,a,a) join(lambda,b,b) x(lambda,0)
not_x(lambda,1) has_state(lambda) not_delta(0,a,0) not_delta(1,a,0)
delta(0,b,0) not_delta(1,b,0) delta(0,a,1) not_delta(1,a,1)
not_delta(0,b,1) not_delta(1,b,1) not_x(a,0) x(b,0) x(a,1)
not_x(b,1) not_final(0) final(1) has_state(a) has_state(b)
```

Note that the above answer set corresponds to a 2-state automaton having non-final state q_0 and final state q_1 (see predicates `not_final(0)` and `final(1)`), and the following transitions $q_0 \in \delta(q_0, b)$, $q_1 \in \delta(q_0, a)$ (defined by predicates `delta(0,b,0)` and `delta(0,a,1)`).

It can be easily verified that $\Pi^X = P$ is the positive program:

```
symbol(a).
symbol(b).
prefix(lambda).
prefix(a).
prefix(b).
positive(a).
negative(b).
join(lambda,a,a).
join(lambda,b,b).
q(0).
q(1).
delta(0,b,0).
delta(0,a,1).
not_delta(0,a,0).
not_delta(1,a,0).
not_delta(1,b,0).
not_delta(1,a,1).
not_delta(0,b,1).
not_delta(1,b,1).
x(lambda,0).
x(b,0).
x(a,1).
not_x(a,0).
not_x(lambda,1).
not_x(b,1).
x(a,0):-delta(1,a,0),x(lambda,1).
x(a,1):-delta(1,a,1),x(lambda,1).
x(b,0):-delta(1,b,0),x(lambda,1).
x(b,1):-delta(1,b,1),x(lambda,1).
x(a,0):-delta(0,a,0),x(lambda,0).
x(a,1):-delta(0,a,1),x(lambda,0).
```

```
x(b,0):-delta(0,b,0),x(lambda,0).
x(b,1):-delta(0,b,1),x(lambda,0).
final(1).
not_final(0).
has_state(lambda):-x(lambda,0).
has_state(a):-x(a,0).
has_state(b):-x(b,0).
has_state(lambda):-x(lambda,1).
has_state(a):-x(a,1).
has_state(b):-x(b,1).
```

$\mathrm{Cn}(P) = X$ since $\bigcup_{i\geq 1} \varnothing^{P^i} = \varnothing^P \cup (\varnothing^P)^P = X$. Further action of P on X does not change it.

References

1. Heinz, J.; de la Higuera, C.; van Zaanen, M. *Grammatical Inference for Computational Linguistics*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2015; Volume 8.
2. Heinz, J.; Sempere, J. (Eds.) *Topics in Grammatical Inference*; Springer-Verlag: Berlin/Heidelberg, Germany, 2016.
3. de la Higuera, C. *Grammatical Inference: Learning Automata and Grammars*; Cambridge University Press: New York, NY, USA, 2010.
4. Miclet, L. Grammatical inference. In *Syntactic and Structural Pattern Recognition: Theory and Applications*; Series in Computer Science; Bunke, H., Sanfeliu, A., Eds.; World Scientific: Singapore, 1990; Volume 7, pp. 237–290.
5. Wieczorek, W. *Grammatical Inference: Algorithms, Routines and Applications*; Studies in Computational Intelligence; Springer: New York, NY, USA, 2017; Volume 673.
6. Meyer, A.R.; Stockmeyer, L.J. The equivalence problem for regular expressions with squaring requires exponential space. In Proceedings of the 13th Annual Symposium on Switching and Automata Theory, College Park, MA, USA, 25–27 October 1972; pp. 125–129.
7. Jiang, T.; Ravikumar, B. Minimal NFA problems are hard. *SIAM J. Comput.* **1993**, *22*, 1117–1141. [CrossRef]
8. Angluin, D. An Application of the Theory of Computational Complexity to the Study of Inductive Inference. Ph.D. Thesis, University of California, Berkeley, CA, USA, 1969.
9. Gold, E.M. Complexity of automaton identification from given data. *Inf. Control.* **1978**, *37*, 302–320. [CrossRef]
10. Domaratzki, M.; Kisman, D.; Shallit, J. On the number of distinct languages accepted by finite automata with *n* states. *J. Autom. Lang. Comb.* **2002**, *7*, 469–486.
11. Jastrzab, T. Two Parallelization Schemes for the Induction of Nondeterministic Finite Automata on PCs. In *PPAM 2017: Parallel Processing and Applied Mathematics*; Wyrzykowski, R., Dongarra, J., Deelman, E., Karczewski, K., Eds.; Springer: Cham, Switzerland, 2018; Volume 10777, pp. 279–289.
12. Heule, M.; Verwer, S. Exact DFA Identification Using SAT Solvers. In *Lecture Notes in Computer Science*; Springer: New York, NY, USA, 2010; Volume 6339, pp. 66–79.
13. Coste, F.; Nicolas, J. Regular Inference as a graph coloring problem. In Proceedings of the Workshop on Grammar Inference, Automata Induction, and Language Acquisition, Nashville, TN, USA, 12 July 1997.
14. Zakirzyanov, I.; Shalyto, A.; Ulyantsev, V. Finding All Minimum-Size DFA Consistent with Given Examples: SAT-Based Approach. In *Software Engineering and Formal Methods*; Cerone, A., Roveri, M., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 117–131.
15. Walsh, T. SAT v CSP. In *Principles and Practice of Constraint Programming—CP 2000*; Dechter, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 441–456.
16. Smetsers, R.; Fiterau-Brostean, P.; Vaandrager, F.W. Model Learning as a Satisfiability Modulo Theories Problem. In Proceedings of the Language and Automata Theory and Applications—12th International Conference, Ramat Gan, Israel, 9–11 April 2018; pp. 182–194.
17. Hopcroft, J.E.; Motwani, R.; Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*, 2nd ed.; Addison-Wesley: Boston, MA, USA, 2001.
18. Baral, C. *Knowledge Representation, Reasoning, and Declarative Problem Solving*; Cambridge University Press: Cambridge, UK, 2003.

19. Gebser, M.; Kaminski, R.; Kaufmann, B.; Schaub, T. *Answer Set Solving in Practice*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2012.
20. Lifschitz, V. *Answer Set Programming*; Springer: New York, NY, USA, 2019.
21. Gebser, M.; Kaufmann, B.; Schaub, T. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.* **2012**, *187*, 52–89. [CrossRef]
22. Gebser, M.; Kaufmann, B.; Schaub, T. Multi-threaded ASP solving with clasp. *arXiv* **2012**, arXiv:1210.3265.
23. Salkind, N.J. *Encyclopedia of Research Design*; SAGE Publications, Inc.: Newbury Park, CA, USA, 2010.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

Article

Parsing Expression Grammars and Their Induction Algorithm

Wojciech Wieczorek [1,*], Olgierd Unold [2] and Łukasz Strąk [3]

[1] Department of Computer Science and Automatics, University of Bielsko-Biala, Willowa 2, 43-309 Bielsko-Biala, Poland
[2] Department of Computer Engineering, Wroclaw University of Science and Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland; olgierd.unold@pwr.edu.pl
[3] Faculty of Science and Technology, University of Silesia in Katowice, Bankowa 14, 40-007 Katowice, Poland; lukasz.strak@us.edu.pl
* Correspondence: wwieczorek@ath.bielsko.pl

Received: 27 October 2020 ; Accepted: 1 December 2020; Published: 7 December 2020

Featured Application: PEG library for Python.

Abstract: Grammatical inference (GI), i.e., the task of finding a rule that lies behind given words, can be used in the analyses of amyloidogenic sequence fragments, which are essential in studies of neurodegenerative diseases. In this paper, we developed a new method that generates non-circular parsing expression grammars (PEGs) and compares it with other GI algorithms on the sequences from a real dataset. The main contribution of this paper is a genetic programming-based algorithm for the induction of parsing expression grammars from a finite sample. The induction method has been tested on a real bioinformatics dataset and its classification performance has been compared to the achievements of existing grammatical inference methods. The evaluation of the generated PEG on an amyloidogenic dataset revealed its accuracy when predicting amyloid segments. We show that the new grammatical inference algorithm achieves the best ACC (Accuracy), AUC (Area under ROC curve), and MCC (Mathew's correlation coefficient) scores in comparison to five other automata or grammar learning methods.

Keywords: classification; genetic programming; grammatical inference; parsing expression grammar

1. Introduction

The present work sits in the scientific field known as grammatical inference (GI), automata learning, grammar identification, or grammar induction [1,2]. The matter under consideration is the set of rules that lie behind a given sequence of words (so-called strings). The main task is to discover the rule(s) that will help us to evaluate new, unseen words. Mathematicians investigate infinite sequences of words and for this purpose they proposed a few inference models. In the most popular model, Gold's identification in the limit [3], learning happens incrementally. After each new word, the algorithm returns some hypothesis, i.e., an automaton or a grammar, and a entire process is regarded as successful when the algorithm returns a correct answer at a certain iteration and does not change it afterwards. However, very often in practice we deal only with a limited number of words (some of them being examples and others counter-examples). In such cases the best option is to use a selected heuristic algorithm, among which the most recognized instances include: evidence driven state merging [4], the *k*-tails method [5], the GIG method [6], the TBL (tabular representation learning) algorithm [7], the learning system ADIOS (automatic distillation of structure) [8], error-correcting grammatical inference [9], and alignment-based learning [10]. However, all of these methods output classical acceptors like (non)deterministic finite state automata (FSA) or context-free grammars (CFG). FSAs are fast in recognition but lack in expressiveness. CFGs, on the other hand, are more expressive but need more computing time for recognizing. We propose here using parsing expression grammars (PEGs), which are as fast as FSAs and can express more than CFGs, in the sense that they can represent some context-sensitive grammars. To the best of our knowledge no one else devised a similar induction algorithm before. As far as non-Chomsky grammars are considered for representing acceptors, Eyraud et al. [11] applied a string-rewriting system to the GI domain. However, as the authors claimed, pure context-sensitive languages can probably not be described with their tool. PEGs are relatively new, but have been implemented in few applications (e.g., Extensible Markup Language schema validation using Document Type Definition automatic transformation [12] and a text pattern-matching tool [13]).

The purpose of the present proposal is threefold. The first objective is to devise an induction algorithm that will suit well real biological data-amyloidogenic sequence fragments. Amyloids are proteins capable of forming fibrils instead of the functional structure of a protein, and are responsible for a group of serious diseases. The second objective is to determine that the proposed algorithm is also well suited for the benchmark data as selected comparative grammatical inference (GI) algorithms and a machine learning approach (SVM). We assume that the given strings do not contain periodically repeated substrings, which is why it has been decided to build up non-circular PEGs that represent finite sets of strings. The last objective is to write a Python library for handling PEGs and make it available to the community. Although there are at least three other Python packages for generating PEG parsers, namely Arpeggio (http://www.igordejanovic.net/Arpeggio), Grako (https://bitbucket.org/neogeny/grako), and pyPEG (https://fdik.org/pyPEG), our implementation (https://github.com/wieczorekw/wieczorekw.github.io/tree/master/PEG) is worth noting for its simple usage (integration with Python syntax via native operators) and because it is dozens of times faster in processing long strings, as will be shown in detail in Section 3.3. In addition to Python libraries, to enrich the research, a library named EGG (https://github.com/bruceiv/egg/tree/deriv) written in C++ was used for comparison, in which an expression has to be compiled into machine code before it is used [14].

This paper is organized into five sections. Section 2 section introduces the notion of parsing expression grammars and also discusses their pros and cons in comparison with regular expressions and CFGs. Section 3 describes the induction algorithm. Section 4 discusses the experimental results. Section 5 summarizes the collected results.

2. Definition of PEGs

PEGs reference *regular expressions* (RE) and *context-free grammars* (CFGs), both derivative from formal language theory. We briefly introduce the most relevant definitions.

An alphabet Σ is a non-empty set of symbols (characters without any meaning). A string or word (s, w) is a finite sequence of symbols. The special case of the string is an empty string ϵ (the empty sequence of symbols). The example of the alphabet is a set $\{a, b, c\}$ and an example of strings over the alphabet is $\{a, aa, ab, ba, abc\}$. A formal language L over an alphabet Σ is a subset of Σ^* (Kleene star, all strings over Σ). A regular expression is a formal way of describing the class of languages called *regular language*. Let r, r_1, and r_2 be the regular expression over Σ, and $a, b \in \Sigma$; the following operations are allowed in syntax:

- ϵ, the empty string;
- a, symbol or string occurrence;
- $r*$, zero or more repetitions of regular expression;
- $a+$, one or more repetitions;
- $a \mid b$, non-deterministic choice of symbol, formally defined as a+b;
- $r_1 r_2$, concatenation;
- (r), parenthesis for grouping of expressions.

Given an alphabet $\Sigma = \{a, b\}$, a formal language $L = \{w \in \Sigma^* \mid w \text{ begins with } a \text{ and ends with } a\}$ can be expressed as the regular expression $a(a \mid b)^* a$. CFG is a tuple of $G = (V, \Sigma, R, S)$, where V is the final set of nonterminal symbols, Σ is the final set of terminal symbols disjoint from V, R is a finite relation $V \rightarrow (V \cup \Sigma)$ and defines rules, and S is the start symbol, chosen from V. The most common R is defined as the production rule notation; for example, for formal language: $L = \{w \in \Sigma^* \mid w^n w^n\ n > 1\}$, the equivalent context-free grammar is $G = (\{S\}, \{a, b\}, P, S)$ with the productions:

$$S \rightarrow aSa$$
$$S \rightarrow bSb$$
$$S \rightarrow a \mid b$$

The word *aba* can be accepted using the first production and the third one. The book by Hopcroft et al. [15] contains more information related to the formal language field.

The formalism of PEGs was introduced by Bryan Ford in 2004 [16]. However, herein we give definitions and notation compatible with the provided PEG library. Let us start with an informal introduction to parsing expression grammars (PEGs).

A *parsing expression grammar* (PEG) is a 4-tuple $G = (V, T, R, s)$, where V is a finite set of nonterminal symbols, T is a finite set of terminal symbols (letters), R is a finite set of rules, s is a parsing expression called the *start expression*, and $V \cap T = \emptyset$. Each rule $r \in R$ is a pair (A, e), which we write as $A \Leftarrow e$, where $A \in V$ and e is a parsing expression. For any nonterminal A, there is exactly one e such that $A \Leftarrow e \in R$. We define *parsing expressions* inductively as follows. If e, e_1, and e_2 are parsing expressions, then so is:

1. ϵ, the empty string;
2. a, any terminal, $a \in T$;
3. A, any nonterminal, $A \in V$;
4. $e_1 \gg e_2$, a sequence;
5. $e_1 \mid e_2$, prioritized choice;
6. $+e$, one or more repetitions;
7. $\sim e$, a not-predicate.

The choice of operators $\gg$, $|$, +, $\sim$, and $\Leftarrow$ is caused by being consistent with our Python library. The operators have their counterparts in Python (`>>`, `|`, `+`, `~`, and `<=`) with the proper precedence. Thus the reader is able to implement expressions in a very natural way, using the native operators.

A PEG is an instance of a recognition system, i.e., a program for recognizing and possibly structuring a string. It can be written in any programming language and looks like a grammar combined with a regex, but its interpretation is different. Take as an example the following regex: $(a \mid b)^+ b$. We can write a "similar" PEG expression: $+(a \mid b) \gg b$. The regex accepts all words over the alphabet $\{a, b\}$ that end with the letter b. The PEG expression, on the contrary, does not recognize any word since PEGs behave greedily, so the part $+(a \mid b)$ will consume all letters, including the last b. An appropriate PEG solution resembles a CFG:

$$A \Leftarrow a \mid b$$
$$E \Leftarrow b \gg \sim A \mid A \gg E$$

The sign $\Leftarrow$ associates an expression to a nonterminal. The sign $\gg$ denotes concatenation. What makes a difference is the ordered choice $|$ and not-predicate $\sim$. The nonterminal E first tries to consume the final b; then, in the case of failure, it consumes a or b and recursively invokes itself. In order to write parsing expressions in a convenient way we will freely omit unnecessary parentheses assuming the following operators precedence (from highest to lowest): $\sim$, +, $\gg$, $|$, $\Leftarrow$. The Kleene star operation can be performed via $+e \mid \epsilon$ (Python does not have a unary star operator and the PEG implementation library had to be adjusted). The power of PEGs is clearly visible in fast, linear-time parsing and in the possibility of expressing some context-sensitive languages [17].

From now on we will use the symbols a, b, and c to represent pairwise different terminals, A, B, C, and D for pairwise different nonterminals, x, x_1, x_2, y, and z for strings of terminals, where $|x_1| = k\,(k \geq 0)$, $|x_2| = m\,(m \geq 0)$, and e, e_1, and e_2 for parsing expressions. To formalize the syntactic meaning of a PEG $G = (V, T, R, s)$, we define a function consume(e, x), which outputs a nonnegative integer (the number of "consumed" letters) or nothing (None):

1. consume$(\epsilon, x) = 0$.
2. consume$(a, ax) = 1$; consume$(a, bx) =$ None; consume$(a, \epsilon) =$ None.
3. consume$(A, x) =$ consume(e, x) if $A \Leftarrow e$.
4. If consume$(e_1, x_1x_2y) = k$ and consume$(e_2, x_2y) = m$, then the following holds: consume$(e_1 \gg e_2, x_1x_2y) = k + m$; if consume$(e_1, x) =$ None, then consume$(e_1 \gg e_2, x) =$ None; if consume$(e_1, x_1y) = k$ and consume$(e_2, y) =$ None, then we can be sure that consume$(e_1 \gg e_2, x_1y) =$ None.
5. If consume$(e_1, x_1y) = k$, then consume$(e_1 \mid e_2, x_1y) = k$; if consume$(e_1, x_1y) =$ None and consume$(e_2, x_1y) = k$, then consume$(e_1 \mid e_2, x_1y) = k$; if consume$(e_1, y) =$ None and consume$(e_2, y) =$ None, then consume$(e_1 \mid e_2, y) =$ None.
6. If consume$(e, x_1y) = k$ and consume$(+e, y) = n$, then consume$(+e, x_1y) = k + n$; if consume$(e, x) =$ None, then consume$(+e, x) =$ None; if consume$(e, x_1y) = k$ and consume$(+e, y) =$ None, then consume$(+e, x_1y) = k$.
7. If consume$(e, x) =$ None, then consume$(\sim e, x) = 0$; if consume$(e, x_1y) = k$, then consume$(\sim e, x_1y) =$ None.

The language $L(G)$ of a PEG $G = (V, T, R, s)$ is the set of strings x for which consume$(s, x) \neq$ None. Please note that the definition of the language of a PEG differs fundamentally from the much more well-known CFGs: in the former it is enough to consume any prefix of a word (including the empty one) to accept it, and in the latter the whole word should be consumed to accept it. Direct (like $A \Leftarrow A \gg e$) as well as indirect left recursions are forbidden, since it can lead to an infinite loop while performing the consume function. It is worth emphasizing that the expression $\sim(\sim e)$ works as non-consuming matching. As a consequence, we can perform language intersection $L(G_1) \cap L(G_2)$ by writing $\sim(\sim s_1) \gg s_2$ if only $G_1 = (V_1, T, R_1, s_1)$, $G_2 = (V_2, T, R_2, s_2)$, and $V_1 \cap V_2 = \emptyset$. Interestingly, it is not proven yet that there exist context-free languages that cannot be recognized by a PEG.

In the next section we deal with non-circular PEGs that will have to be understood as grammars without any recursions or repetitions. Note that such a non-circular PEG, say $G = (\{A\}, T, \{A \Leftarrow e\}, A)$, can be written as a single expression e with no nonterminal and no + operation.

3. Induction Algorithm

The proposed algorithm is based on the genetic programming (GP) paradigm [18]. In it, machine learning can be viewed as requiring discovery of a computer program (an expression in our case) that produces some desired output (the decision class in our case) for particular inputs (strings representing proteins in our case). When viewed in this way, the process of solving problems becomes equivalent to searching a space of possible computer programs for a fittest individual computer program. In this paradigm, populations of computer programs are bred using the principle of survival of the fittest and using a crossover (recombination) operator appropriate for mating computer programs.

This section is split into two subsections. In the first subsection, we will describe the scheme of the GP method adapted to the induction problem. In the second, a deterministic algorithm for the obtaining of an expression matched to the data will be presented. This auxiliary algorithm is used to feed an initial population of GP with promising individuals.

3.1. Genetic Programming

Commonly, genetic programming uses a generational evolutionary algorithm. In generational GP, there exist well-defined and distinct generations. Each generation is represented by a population of individuals. The newer population is created from and then replaces the older population.

The execution cycle of the generational GP—which we used in experiments—includes the following steps:

1. Initialize the population.
2. Evaluate the individual programs in the current population. Assign a numerical fitness to each individual.
3. Until the emerging population is fully populated, repeat the following steps:
   - Select two individuals in the current population using a selection algorithm.
   - Perform genetic operations on the selected individuals.
   - Insert the result of crossover, i.e., the better one out of two children, into the emerging population.
4. If a termination criterion is fulfilled, go to step 5. Otherwise, replace the current population with the emerged population, saving the best individual, and repeat steps 2–4 (elitism strategy).
5. Present the best individual as the output from the algorithm.

In order to put the above procedure to work, we have to define the following elements and routines of GP: the primitives (known in GP as the terminal set and the function set), the structure of an individual, the initialization, genetic operators, and the fitness function.

Individuals are parse trees composed of the PEG's operators $\sim$, $\gg$, and $|$, and terminals are elements of $\Sigma \cup \{\epsilon\}$, where Σ is a finite alphabet (see an example in Figure 1).

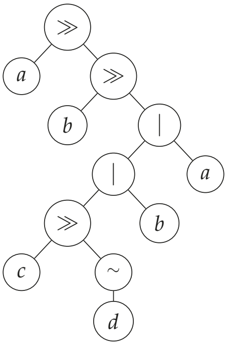


Figure 1. Example of a genetic programming individual coded as the expression $a \gg b \gg (c \gg {\sim}d \mid b \mid a)$.

An initial population is built upon S_+ (positive strings, examples) and S_- (negative strings, counterexamples) so that each tree is consistent with a randomly chosen k-element subset, X, of S_+ and a randomly chosen k-element subset, Y, of S_-. An expression forming an individual in an initial population is created by means of a deterministic algorithm given further on. In a crossover procedure, two expressions given as parse trees are involved. A randomly chosen part of the first tree is replaced by another randomly chosen part from the second tree. The same operation is performed on the second tree in the same manner. We also used tournament selection, in which r (the tournament size) individuals are chosen at random and one of them with the highest fitness is returned. Finally, the fitness function measures an expression's accuracy based on an individual e, and the sample (S_+, S_-) with Equation (1):

$$f(t) = \frac{|\{w \in S_+ : w \in L(G(e))\}| + |\{w \in S_- : w \notin L(G(e))\}|}{|S_+| + |S_-|}, \tag{1}$$

where $G(e)$ is a non-circular PEG $G(e) = (\{A\}, \Sigma, \{A \Leftarrow e\}, A)$.

3.2. Deterministic Algorithm Used in Initializing a GP Population

For a set of strings, S, and a letter, r, by a left quotient, denoted by $r^{-1}S$, we will mean the set $\{w : rw \in S\}$, i.e., $a^{-1}\{ax, ax_1, x_2\} = \{x, x_1\}$. Let X and Y be pairwise disjoint, nonempty sets of words over an alphabet Σ. Our aim is to obtain a compact non-circular PEG G satisfying the following two conditions: (i) $X \subseteq L(G)$, (ii) $Y \cap L(G) = \emptyset$. The Algorithm 1 (function $I(X, Y)$) does it recursively.

Algorithm 1: Inferring a single expression

```
1  function I ($X$, $Y$)
2      $A \leftarrow$ FirstLetters ($X$);                       // set of first letters
3      $B \leftarrow$ FirstLetters ($Y$);
4      Declare $e$ as an empty expression;
5      foreach $a \in A$ do
6          if $a \notin B$ then
7              Append $e$ with $|\ a$ or with $a$ if $e$ is empty;
8          else
9              Append $e$ with $|\ a \gg$ I ($a^{-1}X, a^{-1}Y$);
10     if $e$ is an empty expression then
11         return ~($a \mid b \mid c \mid \ldots$);            // $\{a \mid b \mid c \mid \ldots\} = B$
12     if $X$ has the empty string then
13         Append $e$ with $|$ ~($a \mid b \mid c \mid \ldots) \gg \epsilon$;
14     return $e$;
```

The “Append” method used in lines 7, 9, and 13 in Algorithm 1 concatenates the existing rule e with a new expression. The recursive call I in line 9 cuts sets X and Y to words that start with terminal symbol a and then all words that satisfy this condition are passed with words without the first symbol a (according to the left quotient). Line 10 is used when set A is empty. The execution of the algorithm is shown by the following example. The input is $X = \{abba, bbbb, abaa, abbb, bbaa, bbab\}$, $Y = \{baaa, aaab, babb, aaba, aaaa, baba\}$. Figure 2 shows the successive steps of the algorithm. At the beginning set, A and B are determined. The first symbols in the sets of words X, Y are equal to $\{a, b\}$. The terminal symbol a of the set A belongs to the set B. The string $a \gg$ is added to the rule e and the method is recursively invoked with left quotients $a^{-1}X$ and $a^{-1}Y$ (left leaf from the root in the Figure 2). In the next step, the a symbol is not in the set B. From the recursive call, the b symbol is returned and added to the e rule. After returning, the same procedure is repeated for the symbol b.

The algorithm has the following properties: (i) the X, Y sets in successive calls are always nonempty, (ii) word lengths can be different, (iii) it always halts after a finite number of steps and returns some PEG, (iv) the resultant PEG is consistent with X (examples) and Y (counter-examples), and (v) for random words output PEGs are many times smaller than the input. Properties from (i) to (iii) are quite obvious, (iv) will be proven, and we have checked (v) in a series of experiments, and detailed results are given in the next subsection.

Let $n = |X| + |Y|, m = |\Sigma|$, the length of every word (from X and Y) not exceed d, and T be the running time of the algorithm. Then, for random input words, we can write $T(n, m, d) = O(n) + mT(n/m, m, d-1)$, which leads to $O(m^{\min\{d, \log n\}}n)$. In practice, m and d are small constants, so the running time of $I(X, Y)$ is usually linear with respect to n.

Lemma 1. *Let Σ be a finite alphabet and X, Y be two disjoint, finite, nonempty sets of words over Σ. If $x \in X$ and e is a parsing expression returned by $I(X, Y)$ then consume(e, x) $\neq$ None.*

Proof. We will prove the above by induction on k, where $k \geq 0$ is the length of x.

Basis: We use $k = 0$ as the basis. Because $k = 0$, $x = \epsilon$. Let us consider two cases: (1) ϵ is the only word in X, and (2) $|X| \geq 2$. In the first case, lines 5–9 of the algorithm are skipped and (in line 11)

$e = \sim(a \mid b \mid c \mid \ldots)$ is returned, where $a, b, c, \ldots$ are the first letters of Y (since ϵ is in X, Y has to contain at least one nonempty word). consume($a \mid b \mid c \mid \ldots, \epsilon$) = None implies consume(e, ϵ) = 0. In the second case, the loop in lines 5–9 and line 13 are executed, so the returned expression e (in line 14) has the following form: $\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_j \mid \sim(a \mid b \mid c \mid \ldots) \gg \epsilon$, where α_i is a single letter, say r_i, or $r_i \gg \beta_i$ with β_i being some parsing expression. For such an e, consume(e, ϵ) = 0 holds too.

Induction: Suppose that $|x| = k + 1$ and that the statement of the lemma holds for all words of length j, where $0 \leq j \leq k$. Let $x = uw$, where $u \in \Sigma$. Obviously $|w| = k$. Again let us consider two cases: (1) $u \notin B$, and (2) $u \in B$. In the first case, e, which is returned in line 14, has the form $u \mid \alpha$ or $\alpha \mid u \mid \beta$ or $\alpha \mid u$, where α and β are some expressions. In either case consume(e, uw) ≥ 0 (at least u will not fail for $x = uw$). In the second case, e, which is returned in line 14, is a sequence of addends, one of which is $u \gg \alpha$, where $\alpha = I(u^{-1}X, u^{-1}Y)$. Suffix w is an element of the set $u^{-1}X$ so we invoke the inductive hypothesis to claim that consume(α, w) $\neq$ None. Then consume(e, uw) $\neq$ None, because of the properties of the sequence and the prioritized choice operators (at least $u \gg \alpha$ will not fail for x). □

Lemma 2. *Let Σ be a finite alphabet and X, Y be two disjoint, finite, nonempty sets of words over Σ. If $y \in Y$ and e is a parsing expression returned by $I(X, Y)$ then consume(e, y) = None.*

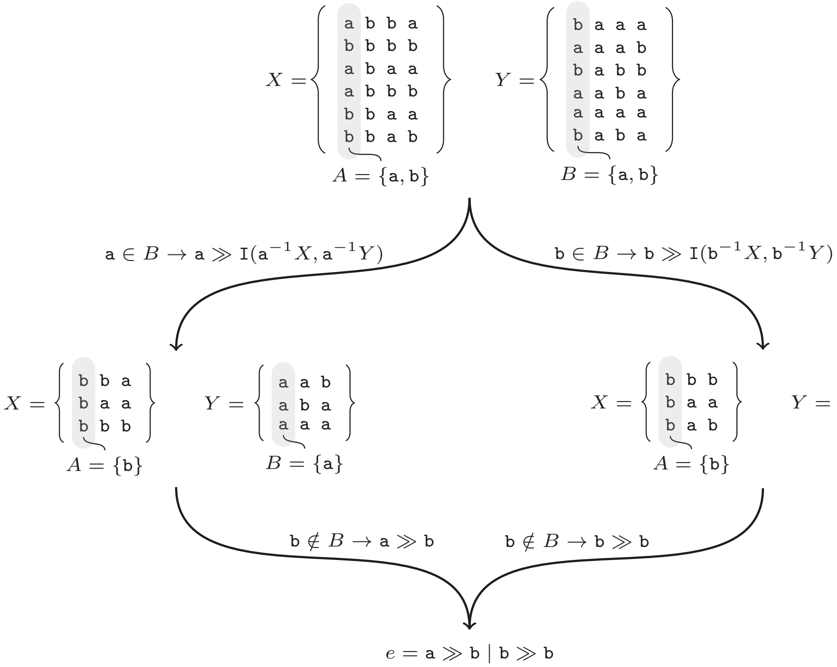


Figure 2. Example of the proposed Induction algorithm.

Proof. We will prove the above by induction on k, where $k \geq 0$ is the length of y.

Basis: We use $k = 0$ as the basis, i.e., $y = \epsilon$. Because $\epsilon \notin X$, the returned (in line 13) expression e has the following form: $\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_j$, where α_i is a single letter, say r_i, or $r_i \gg \beta_i$ with β_i being some parsing expression. For such an e, consume(e, ϵ) = None.

Induction: Suppose that $|y| = k + 1$ and that the statement of the lemma holds for all words of length j, where $0 \leq j \leq k$. Let $y = uw$, where $u \in \Sigma$. Naturally $|w| = k$. There are two main cases to consider: (1) A is empty (that happens only when $X = \{\epsilon\}$), and (2) A is not empty. In the first case, $e = {\sim}(a \mid b \mid c \mid \ldots \mid u \mid \ldots)$ is returned, where $a, b, c, \ldots, u, \ldots$ are the first letters of Y (the position of u in the sequence is not important). consume($a \mid b \mid c \mid \ldots \mid u \mid \ldots, uw) = 1$ implies consume(e, y) $=$ None. In the second case (i.e., A is not empty), let us consider four sub-cases: (2.1) $u \in A$ and $\epsilon \in X$, (2.2) $u \in A$ and $\epsilon \notin X$, (2.3) $u \notin A$ and $\epsilon \in X$, and (2.4) $u \notin A$ and $\epsilon \notin X$. As for (2.1), the returned expression e is of the following form: $\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_j \mid {\sim}(a \mid b \mid c \mid \ldots \mid u \mid \ldots) \gg \epsilon$, where α_i is a single letter, say r_i, or $r_i \gg \beta_i$ with β_i being some parsing expression. Exactly one of α_i has the form $u \gg \beta_i$ (exactly one $r_i = u$), where $\beta_i = I(u^{-1}X, u^{-1}Y)$. Suffix w is an element of the set $u^{-1}Y$ so by the induction hypothesis consume(β_i, w) $=$ None. Then consume(e, uw) $=$ None. Notice that the last addend—i.e., the one with ϵ—will also fail due to consume(${\sim}(a \mid b \mid c \mid \ldots \mid u \mid \ldots), uw$) $=$ None. Sub-case (2.2) is provable similarly to (2.1). When $u \notin A$ (sub-cases 2.3 and 2.4), none of r_i is u and it is easy to see that consume(e, y) $=$ None. □

Theorem 1. *Let Σ be a finite alphabet and X, Y be two disjoint, finite, nonempty sets of words over Σ. If e is a parsing expression returned by $I(X, Y)$ and G is a non-circular PEG defined by $G = (\{A\}, \Sigma, \{A \Leftarrow e\}, A)$ then $X \subseteq L(G)$ and $Y \cap L(G) = \emptyset$.*

Proof. This result follows immediately from the two previous Lemmas. □

3.3. Python's PEG Library Performance Evaluation

In order to assess the fifth property of Algorithm 1 (for random words output PEGs are many times smaller than the input), we created random sets of words with different sizes, lengths and alphabets. Table 1 shows our settings in this respect.

Table 1. The settings of the generator of random input for our PEG algorithm.

No.	$\|\Sigma\|$	$\|X\|$	$\|Y\|$	d_{min}	d_{max}
1	2	10	10	1	10
2	2	100	100	2	20
3	4	500	500	3	30
4	4	1000	1000	4	40
5	8	5000	5000	5	50
6	8	6000	6000	6	60
7	16	7000	7000	7	70
8	16	8000	8000	8	80
9	32	9000	9000	9	90
10	32	10,000	10,000	10	100

Naturally, $|X|$ and $|Y|$ denote the number of examples and counter-examples, while words' lengths vary from d_{max} to d_{min}. Those datasets are publicly available along with the source code of our PEG library. Figure 3 depicts the number of symbols in a PEG and the number of letters in a respective test set.

The number of letters in an input file simply equals $\sum_{w \in X \cup Y} (|w| + 1)$, where $+1$ stands for a new line sign (i.e., words' separator). As for PEGs, the symbol $\gg$ has not been counted, since it may be omitted. Outside the Python language, concatenation of two symbols, for instance a and b, can be written as ab instead of $a \gg b$. Notice also that in Figure 3 the ordinates are in the logarithmic scale, because the differences are large.

The runtime of Python implementation of the proposed PEG library was benchmarked against comparable libraries, i.e., Arpeggio and Grako. The pyPEG library was rejected, because we were unable to define more complex expressions with it. As a testbed we have chosen Tomita's languages [19]. This test set contains seven different expressions that serve as rules for the generation of words over a binary alphabet. Their description in a natural language can be found in Table 2. Seven regular expressions appropriate to the rules were created, and then the generators of random input words were implemented. Thus, for every language we had two sets: matching (positive) and non-matching (negative) words to a particular regular expression. These expressions take the following forms:

1. a*
2. (ab)*
3. ((b|(aa))|(((a(bb))((bb)|(a(bb)))*)(aa)))*((a?)|(((a(bb))((bb)|(a(bb)))*)(a?)))
4. a*((b|bb)aa*)*(b|bb|a*)
5. (aa|bb)*((ba|ab)(bb|aa)*(ba|ab)(bb|aa)*)*(aa|bb)*
6. ((a(ab)*(b|aa))|(b(ba)*(a|bb)))*
7. a*b*a*b*

Equivalent PEG expressionswere defined as well in every comparable library (see Table 2).

Table 2. PEG expressions created based on *Tomita* (1982) languages.

No.	Description	PEG Grammar
1	Sequence of a's	$S \Leftarrow a \gg S \mid \sim(a \mid b)$
2	Sequence of (ab)'s	$S \Leftarrow a \gg b \gg S \mid \sim(a \mid b)$
3	Any string without an odd number of consecutive b's after an odd number of consecutive a's	$A \Leftarrow a \gg \sim a \mid a \gg a \gg A$ $B \Leftarrow b \gg \sim b \mid b \gg b \gg B$ $C \Leftarrow a \gg A \mid \sim a$ $D \Leftarrow b \gg B \mid \sim b$ $S \Leftarrow +(a \gg a \mid b) \gg S \mid A \gg D \gg S \mid \sim(a \mid b)$
4	Any string without more than two consecutive b's	$S \Leftarrow (+a \mid \epsilon) \gg (+((b \gg b \mid b) \gg +a) \mid \epsilon) \gg$ $(b \gg b \gg \sim(a \mid b) \mid b \gg \sim(a \mid b) \mid \sim(a \mid b))$
5	Any string of even length that, making pairs, has an even number of (ab)'s or (ba)'s	$A \Leftarrow +(a \gg a \mid b \gg b) \mid \epsilon$ $B \Leftarrow a \gg b \mid b \gg a$ $S \Leftarrow A \gg (+(B \gg A \gg B \gg A) \mid \epsilon) \gg A \gg \sim(a \mid b)$
6	Any string such that the difference between the numbers of a's and b's is a multiple of three	$A \Leftarrow a \gg (+(a \gg b) \mid \epsilon) \gg (b \mid a \gg a)$ $B \Leftarrow b \gg (+(b \gg a) \mid \epsilon) \gg (a \mid b \gg b)$ $S \Leftarrow +(A \mid B) \gg \sim(a \mid b) \mid \sim(a \mid b)$
7	Zero or more a's followed by zero or more b's followed by zero or more a's followed by zero or more b's	$A \Leftarrow +a \mid \epsilon$ $B \Leftarrow +b \mid \epsilon$ $S \Leftarrow A \gg B \gg A \gg B \gg \sim(a \mid b)$

Table 3 summarizes CPU time results. Every row contains the means for 30 runs. In all experiments we used the implementation of algorithms written in Python (our PEG library, Grako, and Arpeggio) and C++ EGG. An interpreter ran on a four-core Intel i7-965, 3.2 GHz processor in a Windows 10 operating system with 12 GB RAM.

As can be seen, in all cases our library worked much faster than other Python libraries. Grammar 3 was skipped because we were unable to define it either by means of the Arpeggio or Grako libraries. It should be stated, however, that both of the libraries have more functionality than our PEG library, its principal function being only the membership operation, i.e., matching or not a word to a PEG. As a result, Arpeggio, Grako, and pyPeg are relatively not intuitive and obvious, especially for users not familiarized with formal languages theory. The dash character in the EGG result denotes segmentation runtime error. As expected the C++ library (EGG) overcame its Python counterparts.

Table 3. Average CPU times for available Python PEG libraries (in seconds).

Case No.	Positive/Negative	Word Length	PEG [s]	Grako [s]	Arpeggio [s]	EGG [s]
1	Positive	1–100	0.01	0.21	0.02	<0.01
1	Positive	101–1000	0.04	1.93	0.13	<0.01
1	Positive	1001–10,000	0.44	17.62	0.92	0.01
1	Positive	10,001–100,000	5.93	182	9.82	0.01
1	Negative	1–100	<0.01	0.11	0.01	0.06
1	Negative	101–1000	0.02	1.13	0.05	0.02
1	Negative	1001–10,000	0.24	9.68	0.42	–
1	Negative	10,001–100,000	2.41	81.95	3.58	–
2	Positive	1–100	0.01	0.19	0.01	<0.01
2	Positive	101–1000	0.03	1.68	0.12	<0.01
2	Positive	1001–10,000	0.12	5.64	0.4	<0.01
2	Positive	10,001–100,000	1.2	47.95	3.38	<0.01
2	Negative	1–100	<0.01	0.11	0.01	0.01
2	Negative	101–1000	0.02	0.82	0.05	0.01
2	Negative	1001–10,000	0.07	3.19	0.18	0.08
2	Negative	10,001–100,000	0.59	24.54	1.42	0.05
4	Positive	1–100	<0.01	0.2	0.02	<0.01
4	Positive	101–1000	0.04	2.12	0.36	<0.01
4	Positive	1001–10,000	0.26	12.37	2.28	<0.01
4	Positive	10,001–100,000	2.40	102.19	20.29	0.01
4	Negative	1–100	0.01	0.27	0.02	0.03
4	Negative	101–1000	0.04	2.11	0.32	0.03
4	Negative	1001–10,000	0.25	11.93	2.01	0.22
4	Negative	10,001–100,000	2.32	98.61	17.98	0.22
5	Positive	1–100	<0.01	0.2	0.02	<0.01
5	Positive	101–1000	0.04	2.12	0.36	<0.01
5	Positive	1001–10,000	0.26	12.37	2.28	0.01
5	Positive	10,001–100,000	2.4	102.19	20.29	<0.01
5	Negative	1–100	0.01	0.27	0.02	0.05
5	Negative	101–1000	0.04	2.11	0.32	0.02
5	Negative	1001–10,000	0.25	11.93	2.01	0.08
5	Negative	10,001–100,000	2.32	98.61	17.98	0.08
6	Positive	1–100	0.01	0.23	0.02	<0.01
6	Positive	101–1000	0.05	2.52	0.17	<0.01
6	Positive	1001–10,000	0.40	17.64	1.21	<0.01
6	Positive	10,001–100,000	1.67	71.70	4.93	0.01
6	Negative	1–100	0.01	0.34	0.02	0.03
6	Negative	101–1000	0.12	5.34	0.33	0.06
6	Negative	1001–10,000	1.05	44.86	2.61	0.09
6	Negative	10,001–100,000	4.61	187.90	10.42	0.17
7	Positive	1–100	<0.01	0.23	0.02	<0.01
7	Positive	101–1000	0.04	1.85	0.09	<0.01
7	Positive	1001–10,000	0.17	7.94	0.42	0.01
7	Positive	10,001–100,000	1.91	73.4	3.92	<0.01
7	Negative	1–100	<0.01	0.16	0.01	0.03
7	Negative	101–1000	0.02	1.09	0.06	0.02
7	Negative	1001–10,000	0.11	5.37	0.28	0.15
7	Negative	10,001–100,000	1.26	49.92	2.64	0.10

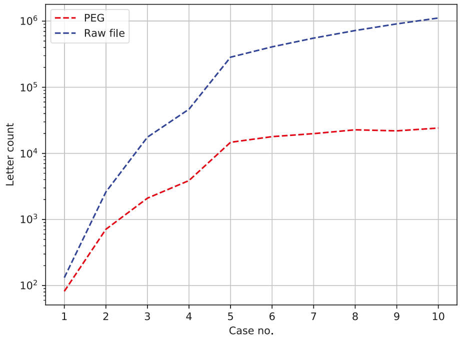


Figure 3. The number of symbols in a PEG (red line) and the number of letters in a respective test set (blue line).

4. Results and Discussion

The algorithm for generating non-circular parsing expression grammars (PEG) was tested over a recently published amyloidogenic dataset [20]. The GP parameters (John Koza, a GP pioneer, has introduced a very lucid form of listing parameters in the tableau of Table 4 named after him) are listed in Table 4. From there, we can read that a population size of $P = 5$ individuals were used for GP runs along with others. The terminal set contains standard amino acid abbreviations; "A" stands for Alanine, "R" for Arginine, etc. Concerning the initialization method, see Section 3.2. The best parameters were chosen in a trial-and-error manner until the values with the best classification quality were found. The dataset is composed of 1476 strings that represent protein fragments. The data came from four databases as shown in Figures 4 and 5. A total of 439 are classified as being amyloidogenic (examples), and 1037 as not (counter-examples). The shortest sequence length is 4 and the longest is 83. Such a wide range of sequence lengths was an additional impediment to learning algorithms.

Table 4. Koza tableau.

Parameters	Values
Objective:	evolve expression classifying amino acid sequences according to examples and counterexamples
Terminal set:	ϵ, A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V
Function set:	$\sim, \gg, \mid$
Population size:	5
Crossover probability:	1.0
Selection:	Tournament selection, size $r = 3$
Termination criterion:	6000 generations have passed
Maximum depth of tree after crossover:	100
Initialization method:	A special dedicated algorithm, $k = \lvert X \rvert = \lvert Y \rvert$ equals half of the cardinality of examples

In order to compare our algorithm to other grammatical inference approaches, we took most of the methods mentioned in the introductory section as a reference. Error-correcting grammatical inference [9] (ECGI) and alignment-based learning [10] (ABL) are examples of substring-based algorithms. The former builds an automaton incrementally based on the Levenstein distance between the closest word stored in the automaton and an inserted word. This process begins with an empty automaton, and for each word adds the error rules (insertion, substitution, and deletion) belonging to the transition path with the least number of error rules. The algorithm provides an automaton without loops that is more and more general. The latter, ABL, is based on searching identical and distinct parts of input words. This algorithm consists of two stages. First, all words are aligned such that it finds a shared and a distinct part of all pairs of words, suggesting that the distinct parts have the same type. For example, consider the pair "abcd" and "abe". Here, "cd" and "e" are correctly identified as examples of the same type. The second step, which takes the same corpus as input, tries to identify the right constituents. Because the generated constituents found in the previous step might overlap, the correct ones have to be selected. Simple heuristics are used to achieve this, for example to take the constituent that was generated first (ABL-first) or to take the constituent with the highest score on some probabilistic function. We used another approach, in which all constituents are stored, but in the end we tried to keep only the minimum number of constituents that cover all examples.

ADIOS uses statistical information present in sequential data to identify significant segments and to distill rule-like regularities that support structured generalization [8]. It also brings together several crucial conceptual components; the structures it learns are (i) variable-order, (ii) hierarchically composed, (iii) context dependent, (iv) supported by a previously undocumented statistical significance criterion, and (v) dictated solely by the corpus at hand.

Blue-fringe [21] and Traxbar [22], the instances of state merging algorithms, can be downloaded from an internet archive (http://abbadingo.cs.nuim.ie/dfa-algorithms.tar.gz). They start from building a prefix tree acceptor (PTA) based on examples, and then iteratively select two states and do merging unless compatibility is broken. The difference between them comes from many ways in which the pair of states needed to merge can be chosen. Trakhtenbrot and Barzdin [23] described an algorithm for constructing the smallest deterministic FSA consistent with a complete labeled training set. The PTA is squeezed into a smaller graph by merging all pairs of states that represent compatible mappings from word suffixes to labels. This algorithm for completely labeled trees was generalized by Lang (1992) [22] to produce a (not necessarily minimum) automaton consistent with a sparsely labeled tree. Blue-fringe grows a connected set of red nodes that are known to be unique states, surrounded by a fringe of blue nodes that will either be merged with red nodes or promoted to red status. Merges only occur between red nodes and blue nodes. Blue nodes are known to be the roots of trees, which greatly simplifies the code for correct merging.

We also included one machine learning approach. An unsupervised data-driven distributed representation, called ProtVec [24], was applied and protein family classification was performed using a support vector machine classifier (SVM) [25] with the linear kernel.

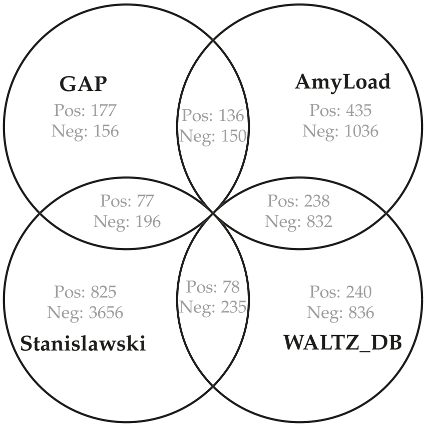


Figure 4. Combined amyloid databases used in this work. Pos and Neg denote, respectively, positive and negative word counts in the database.

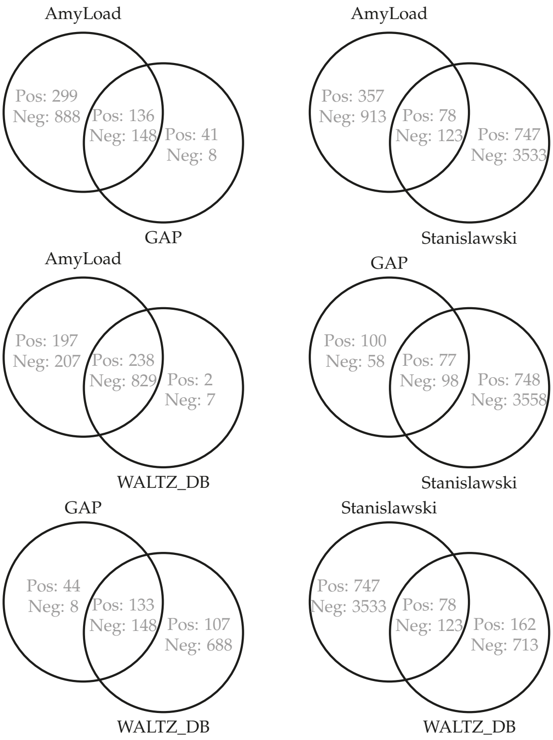


Figure 5. Combined amyloid databases used in work. Pos and Neg denote, respectively, positive and negative word counts in the database.

The data were randomly split into two subsets, a training (75% of total) and a test set (25% of total). Given the training set and the test set, we used all algorithms to infer predictors (automata or grammars) on the training set, tested them on the test set, and computed their performances. Comparative analyses of the following five measures: Precision, Recall, F-score, the AUC, and Matthews correlation coefficient are summarized in Table 5. The measures are given below:

- Precision, $\mathrm{P} = tp/(tp + fp)$;
- Recall, $\mathrm{R} = tp/(tp + fn)$;
- F-score, $\mathrm{F1} = 2 \times \mathrm{P} \times \mathrm{R}/(\mathrm{P} + \mathrm{R})$;
- Accuracy, $\mathrm{ACC} = (tp + tn)/(tp + tn + fp + fn)$;
- Area under the ROC curve, $\mathrm{AUC} = (tp/(tp + fn) + tn/(fp + tn))/2$;
- Matthews correlation coefficient, $\mathrm{MCC} = \frac{tp\times tn - fp\times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$;

where the terms true positives (*tp*), true negatives (*tn*), false positives (*fp*), and false negatives (*fn*) compare the results of the classifier under test with trusted external judgments. Thus, in our case, *tp* is the number of correctly recognized amyloids, *fp* is the number of nonamyloids recognized as amyloids, *fn* is the number of amyloids recognized as nonamyloids, and *tn* is the number of correctly recognized nonamyloids. The last column concerns CPU time of computations (induction plus classification in s).

Table 5. Results of classification quality for the test set by the decreasing AUC.

	P	R	F1	ACC	AUC	MCC	Time [s]
PEG	0.627	0.294	0.400	0.739	0.610	0.291	0.3
ABL	0.645	0.183	0.286	0.728	0.571	0.232	412.9
ADIOS	0.329	0.633	0.433	0.508	0.544	0.082	15.6
Blue-fringe	0.367	0.303	0.332	0.639	0.541	0.088	0.9
ECGI	0.875	0.064	0.120	0.720	0.530	0.189	31.0
Traxbar	0.234	0.101	0.141	0.636	0.481	−0.05	0.3
SVM	0.224	0.001	0.131	0.526	0.471	−0.06	0.4

The results show that there is no single method that outperformed the remaining methods regardless of an established classification measure. However, the methods can be grouped as relatively good and relatively weak from a certain angle. As regards Recall and F-score, relatively good are Blue-fringe, ADIOS, and PEG. As regards MCC, which is generally recognized as being one of the best classification measures, relatively good are PEG and ABL. Moreover, PEG achieved the best AUC, which in the case of binary prediction is equivalent to balanced accuracy.

To evaluate the convergence toward an optimal solution, we studied average fitness change over generations along with the increasing of the expression sizes (see Figure 6). The shape of the plot does not show any indication of premature convergence. Moreover, we did not observe excessive tree size expansion, which is quite often seen in genetic programming.

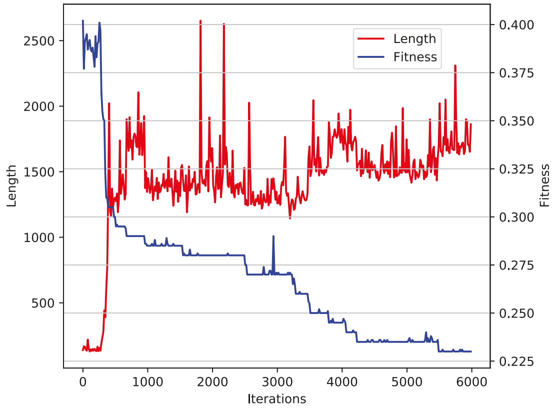


Figure 6. Average error (1−fitness accuracy) vs. expression length for different generations based on random data with two letters in the alphabet, 100 words at each set of example and counter-example and word lengths between 2 and 20.

All programs ran on an Intel Xeon CPU E5-2650 v2, 2.6 GHz processor under an Ubuntu 16.04 operating system with 192 GB RAM. The computational complexity of all the algorithms is polynomially bounded; however, the differences in running time were quite significant and our approach ranked at the top. The algorithm for PEG induction (https://github.com/wieczorekw/wieczorekw.github.io/tree/master/PEG) was written in the Python 3 programming language. The languages of implementation for six successive methods were: Python, Java, C, Python, C, and Python.

5. Conclusions

We proposed a new grammatical inference (PEG-based) method and applied it to a real bioinformatics task, i.e., classification of amyloidogenic sequences. The evaluation of generated PEGs on an amyloidogenic dataset revealed the method's accuracy in predicting amyloid segments. We showed that the new grammatical inference algorithm gives the best ACC, AUC, and MCC scores in comparison to five other automata or grammar learning methods and the ProtVec/SVM method.

In the future, we will implement circular rules in the PEG library, which will improve the expressiveness of grammars and may improve the quality of classification.

Author Contributions: Conceptualization, W.W.; formal analysis, W.W., O.U. and Ł.S.; investigation, W.W. and Ł.S.; methodology, W.W. and O.U.; software W.W., Ł.S.; validation, W.W. and Ł.S.; writing—original draft, W.W. and Ł.S.; writing—review & editing, W.W., O.U., Ł.S.; data curation, O.U.; funding acquisition, O.U.; project administration, O.U.; supervision, O.U.; resources, Ł.S.; visualization, Ł.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Science Center, grant 2016/21/B/ST6/02158.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. De la Higuera, C. *Grammatical Inference: Learning Automata and Grammars*; Cambridge University Press: New York, NY, USA, 2010.
2. Wieczorek, W. *Grammatical Inference—Algorithms, Routines and Applications*; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2017; Volume 673. [CrossRef]
3. Gold, E.M. Language Identification in the Limit. *Inf. Control* **1967**, *10*, 447–474. [CrossRef]
4. Coste, F.; Fredouille, D. Unambiguous Automata Inference by Means of State-Merging Methods. In Proceedings of the Machine Learning: ECML 2003, 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, 22–26 September 2003; pp. 60–71.
5. Miclet, L. Grammatical Inference. In *Syntactic and Structural Pattern Recognition Theory and Applications*; Bunke, H., Sanfeliu, A., Eds.; World Scientific Series in Computer Science; World Scientific: Singapore, 1990; Volume 7, pp. 237–290.
6. Dupont, P. Regular grammatical inference from positive and negative samples by genetic search: The GIG method. In *Grammatical Inference and Applications*; Carrasco, R.C., Oncina, J., Eds.; Springer: Berlin/Heidelberg, Germany, 1994; pp. 236–245.
7. Sakakibara, Y. Learning context-free grammars using tabular representations. *Pattern Recognit.* **2005**, *38*, 1372–1383. Grammatical Inference, doi:10.1016/j.patcog.2004.03.021. [CrossRef]
8. Solan, Z.; Horn, D.; Ruppin, E.; Edelman, S. Unsupervised learning of natural languages. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 11629–11634, [CrossRef] [PubMed]
9. Rulot, H.; Vidal, E. Modelling (Sub)String Length Based Constraints through a Grammatical Inference Method. In *Proceedings of the NATO Advanced Study Institute on Pattern Recognition Theory and Applications*; Devijver, P.A., Kittler, J., Eds.; Springer: Berlin/Heidelberg, Germany, 1987; pp. 451–459.
10. van Zaanen, M. ABL: Alignment-based Learning. In Proceedings of the 18th Conference on Computational Linguistics—Volume 2, Jcken, Germany, 31 July–4 August 2000; Association for Computational Linguistics: Stroudsburg, PA, USA, 2000; COLING '00, pp. 961–967. [CrossRef]
11. Eyraud, R.; de la Higuera, C.; Janodet, J.C. LARS: A learning algorithm for rewriting systems. *Mach. Learn.* **2007**, *66*, 7–31. [CrossRef]
12. Kuramitsu, K.; ya Hamaguchi, S. XML schema validation using parsing expression grammars. *PeerJ Prepr.* **2015**, *3*, e1503.
13. Ierusalimschy, R. A text pattern-matching tool based on Parsing Expression Grammars. *Softw. Pract. Exp.* **2009**, *39*, 221–258. [CrossRef]
14. Moss, A. Simplified Parsing Expression Derivatives. In *Language and Automata Theory and Applications*; Leporati, A., Martín-Vide, C., Shapira, D., Zandron, C., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 425–436.

15. Hopcroft, J.E.; Motwani, R.; Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*, 2nd ed.; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2001.
16. Ford, B. Parsing expression grammars: A recognition-based syntactic foundation. In Proceedings of the 31st ACM SIGACT/SIGPLAN Symposium on Principles of Programming Languages, Venice, Italy, 14–16 January 2004; Volume 39, pp. 111–122.
17. Grune, D.; Jacobs, C. *Parsing Techniques: A Practical Guide*; Springer: New York, NY, USA, 2008; pp. 1–662. [CrossRef]
18. Banzhaf, W.; Francone, F.D.; Keller, R.E.; Nordin, P. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1998.
19. Tomita, M. *Learning of Construction of Finite Automata from Examples Using Hill-Climbing: RR: Regular Set Recognizer*; Department of Computer Science, Carnegie-Mellon University: Pittsburgh, PA, USA, 1982.
20. Wozniak, P.; Kotulska, M. AmyLoad: Website dedicated to amyloidogenic protein fragments. *Bioinformatics* **2015**, *31*, 3395–3397. [CrossRef] [PubMed]
21. Lang, K.; Pearlmutter, B.; Price, R. *Results of the Abbadingo one DFA Learning Competition and a New Evidence-Driven State Merging Algorithm*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 1998; Volume 1433, pp. 1–12.
22. Lang, K.J. Random DFA's can be approximately learned from sparse uniform examples. In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; ACM: New York, NY, USA; Pittsburgh, PA, USA, 1992; pp. 45–52.
23. Trakhtenbrot, B.; Barzdin, Y. *Finite Automata: Behavior and Synthesis*; Fundamental Studies in Computer Science; North-Holland Publishing Company: Amsterdam, The Netherlands, 1973.
24. Asgari, E.; Mofrad, M.R.K. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLoS ONE* **2015**, *10*, 1–15. [CrossRef] [PubMed]
25. Wu, T.F.; Lin, C.J.; Weng, R.C. Probability Estimates for Multi-class Classification by Pairwise Coupling. *J. Mach. Learn. Res.* **2004**, *5*, 975–1005.

applied sciences

MDPI

Article

Anticipatory Classifier System with Average Reward Criterion in Discretized Multi-Step Environments

Norbert Kozłowski [*,†] **and Olgierd Unold** [†]

Department of Computer Engineering, Faculty of Electronics, Wroclaw University of Science and Technology, 50-370 Wroclaw, Poland; olgierd.unold@pwr.edu.pl

[*] Correspondence: norbert.kozlowski@pwr.edu.pl; Tel.: +48-792-922-331

[†] These authors contributed equally to this work.

Abstract: Initially, Anticipatory Classifier Systems (ACS) were designed to address both single and multistep decision problems. In the latter case, the objective was to maximize the total discounted rewards, usually based on Q-learning algorithms. Studies on other Learning Classifier Systems (LCS) revealed many real-world sequential decision problems where the preferred objective is the maximization of the average of successive rewards. This paper proposes a relevant modification toward the learning component, allowing us to address such problems. The modified system is called AACS2 (Averaged ACS2) and is tested on three multistep benchmark problems.

Keywords: learning classifier systems; anticipatory classifier systems; reinforcement learning; genetic algorithms; OpenAI gym

Citation: Kozłowski, N.; Unold, O. Anticipatory Classifier System with Average Reward Criterion in Discretized Multi-Step Environments. *Appl. Sci.* **2021**, *11*, 1098. https://doi.org/10.3390/app11031098

Academic Editor: Grzegorz Dudek
Received: 28 October 2020
Accepted: 16 January 2021
Published: 25 January 2021

1. Introduction

Learning Classifier Systems (LCS) [1] comprise a family of flexible, evolutionary, rule-based machine learning systems that involve a unique tandem of local learning and global evolutionary optimization of the collective model localities. They provide a generic framework combining the discovery and learning components. Despite the misleading name, LCSs are not only suitable for classification problems but may instead be viewed as a very general, distributed optimization technique. Due to representing knowledge locally as IF-THEN rules with additional parameters (such as predicted payoff), they have high potential to be applied in any problem domain that is best solved or approximated through a distributed set of local approximations or predictions. The main feature of LCS is the employment of two learning components. The discovery mechanism uses the evolutionary approach to optimize the individual structure of each classifier. On the other side, there is a credit assignment component approximating the classifier fitness estimation. Because those two interact bidirectionally, LCSs are often perceived as being hard to understand.

Nowadays, LCS research is moving in multiple directions. For instance, BioHEL [2] and ExSTraCS [3] algorithms are designed to handle large amounts of data. They extend the basic idea by adding expert-knowledge-guided learning, attribute tracking for heterogeneous subgroup identification, and a number of other heuristics to handle complex and noisy data mining. On the other side, there are some advances made towards combining LCS with artificial neural networks [4]. Liang et al. [5] took the approach of combining the feature selection of *Convolutional Neural Networks* with LCSs. Tadokoro et al. [6] have a similar goal—they want to use *Deep Neural Networks* for preprocessing in order to be able to use LCSs for high-dimensional data while preserving their inherent interpretability. An overview of recent LCS advancements is published yearly as a part of the *International Workshop on Learning Classifier Systems* (IWLCS) [7].

In the most popular LCS modification-XCS [8], where the classifier fitness is based on the *accuracy* of a classifier's payoff prediction instead of the prediction itself, the learning component responsible for local optimization follows the Q-learning [9] pattern. Classifier predictions are updated using the immediate reward and the discounted maximum payoff anticipated in the next time step. The difference is that, in XCS, it is the prediction of a general rule that is updated, whereas, in Q-learning, it is the prediction associated with an environmental *state–action* pair. In this case, both algorithms are suitable for multistep (sequential) decision problems in which the objective is to maximize the discounted sum of rewards received in successive steps.

However, in many real-world situations, the discounted sum of rewards is not the appropriate option. This choice is right when the rewards received in all decision instances are equally important. The criterion applied in this situation is called *the average reward criterion* and was introduced by Puterman [10]. He stated that the decision maker might prefer it when the decisions are made frequently (so that the discount rate is very close to 1) or other terms cannot easily describe the performance criterion. Possible areas of an application might include situations where system performance is assessed based on the throughput rate (like making frequent decisions when controlling the flow of communication networks).

The averaged reward criterion was first implemented in XCS by Tharakunnel and Goldberg [11]. They called their modification AXCS and showed that it performed similarly to the standard XCS in the Woods2 environment. Later, Zang et al. [12] formally introduced the R-learning [13,14] technique to XCS and called it XCSAR. They compared it with XCSG (where the prediction parameters are modified by applying the idea of gradient descent) and ACXS (maximizing the average of successive rewards) in large multistep problems (Woods1, Maze6, and Woods14).

In this paper, we introduce the average reward criterion to yet another family of LCSs-anticipatory learning classifier systems (ALCS). They differentiate from others so that a predictive schema model of the environment is learned rather than reward prediction maps. In contrast to the usual classifier structure, classifiers in ALCS have a state prediction or *an anticipatory part* that predicts the environmental changes caused when executing the specified action in the specified context. Similarly, as in the XCS, ALCSs derive classifier fitness estimates from the accuracy of their predictions; however, anticipatory state predictions' accuracy is considered rather than the reward prediction accuracy. Popular ALCSs use the discounted criterion in the original form, optimizing the performance in the infinite horizon.

Section 2 starts by briefly describing the psychological insights from the concepts of imprinting and anticipations and the most popular ALCS architecture-ACS2. Then, the RL and the ACS2 learning components are described. The default discounted reward criterion is formally defined, and two versions of undiscounted (averaged) criterion integration are introduced. The created system is called AACS2, which stands for *Averaged ACS2*. Finally, three testing sequential environments with increasing difficulty are presented: the Corridor, Finite State Worlds, and Woods. Section 3 examines and describes the results of testing ACS2, AACS2, Q-learning, and R-learning in all environments. Finally, the conclusions are drawn in Section 4.

2. Materials and Methods

2.1. Anticipatory Learning Classifier Systems

In 1993, Hoffman proposed a theory of *"Anticipatory Behavioral Control"* that was further refined in [15]. It states that higher animals form an internal environmental representation and adapt their behavior through learning anticipations. The following points (visualized in Figure 1) can be distinguished:

1. Any behavioral act or response (R) is accompanied by anticipation of its effects.
2. The anticipations of the effects E_{ant} are compared with the real effects E_{real}.
3. The bond between response and anticipation is strengthened when the anticipations were correct and weakened otherwise.
4. Behavioral stimuli further differentiate the $R - E_{ant}$ relations.

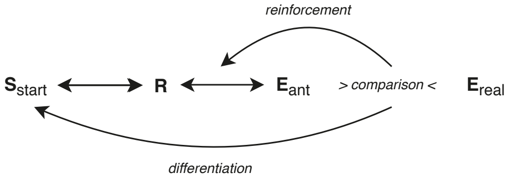


Figure 1. The theory of anticipatory behavioral control; the figure was adapted from [16], p. 4.

This insight into the presence and importance of anticipations in animals and man leads to the conclusion that it would be beneficial to represent and utilize them in animals.

Stolzmann took the first approach in 1997 [17]. He presented a system called ACS (*"Anticipatory Classifier System"*), enhancing the classifier structure with an anticipatory (or effect) part that anticipates the effects of an action in a given situation. A dedicated component realizing Hoffmann's theory was proposed—*Anticipatory Learning Process* (ALP), introducing new classifiers into the system.

The ACS starts with a population $[P]$ of most general classifiers ('#' in a condition part) for each available action. To ensure that there is always a classifier handling every consecutive situation, those cannot be removed. During each behavioral act, the current perception of environment $\sigma(t)$ is captured. Then, a match set $[M](t)$ is formed, consisting of all classifiers from $[P]$ where the condition matches the perception $\sigma(t)$. Next, one classifier *cl* is drawn from $[M](t)$ using some exploration policy. Usually, an epsilon-greedy technique is used, but [18] describes other options as well. Then, the classifier's action $cl.a$ is executed in the environment, and a new perception $\sigma(t+1)$ and reward $\phi(t+1)$ values are presented to the agent. Knowing the classifiers' anticipation and current state, the ALP module can adjust the classifier cl's condition and effect parts. Based on this comparison, certain cases might be present:

- *Useless-case*. After performing an action, no change in perception is perceived from the environment. The classifier's quality $cl.q$ decreases.
- *Unexpected-case*. When new state $\sigma(t+1)$ does not match the anticipation of $cl.E$. A new classifier with a matching effect part is generated, and the incorrect one is penalized with a quality drop.
- *Expected-case*. When the newly observed state matches the classifier prediction. Classifiers' quality is increased.

After the ALP application, the Reinforcement Learning (RL) module is executed (see Section 2.3 for details).

Later, in 2002, Butz presented an extension to the described system called ACS2 [16]. Most importantly, he modified the original approach by:

1. explicit representation of anticipations,
2. applying learning components across the whole action set $[A]$ (all classifiers from $[M]$ advocating selected action),
3. introduction of *Genetic Generalization* module for generating new classifier using promising offsprings,
4. changing the RL module motivated by the Q-Learning algorithm.

Figure 2 presents the complete behavioral act, and Refs. [19,20] describe the algorithm thoroughly.

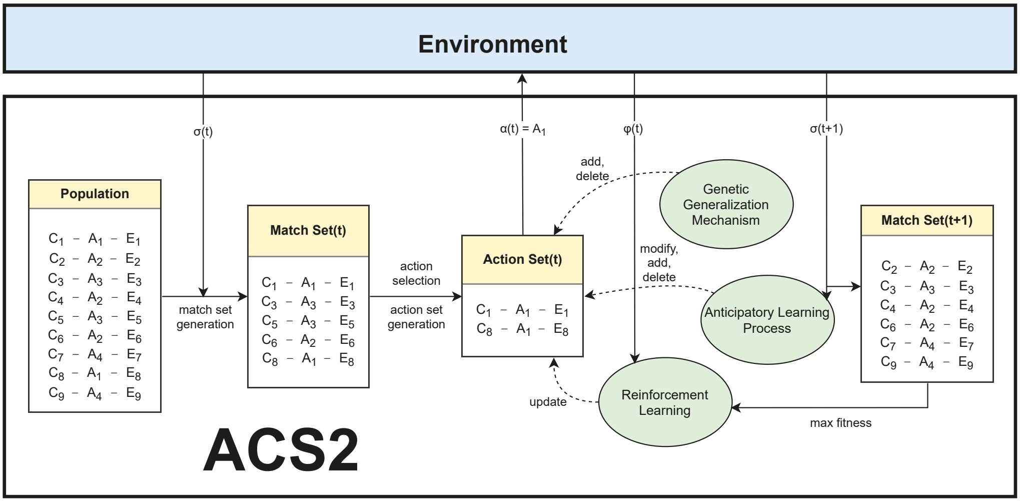


Figure 2. A behavioral act in ACS2; the figure was adapted from [16], p. 27.

Some modifications were made later to the original ACS2 algorithm. Unold et al. integrated the action planning mechanism [21], Orhand et al. extended the classifier structure with *Probability-Enhanced-Predictions* introducing a system capable of handling non-deterministic environments and calling it PEPACS [22]. In the same year, they also tackled an issue of perceptual aliasing by building a *Behavioral Sequences*—thus creating a system called BACS [23].

2.2. Reinforcement Learning and Reward Criterion

Reinforcement Learning (RL) is a formal framework in which the agent can influence the environment by executing specific actions and receive corresponding feedback (reward) afterwards. Usually, it takes multiple steps to reach the goal, which makes the process much more complicated. In the general form, RL consists of:

- A discrete set of environment states S,
- A discrete set of available actions A,
- A mapping R between a particular state $s \in S$ and action $a \in A$. The environmental payoff $r \in R$ describes the expected reward obtained after executing an action in a given state.

In each trial, the agent perceives the environmental state s. Next, it evaluates all possible actions from A and executes action a in the environment. The environment returns a signal r and next state s' as intermediate feedback.

The agent's task is to represent the knowledge, using the policy π mapping states to actions, therefore optimizing a long-run measure of reinforcement. There are two popular optimality criteria used in Markov Decision Problems (MDP)—a *discounted reward* and an *average reward* [24,25].

2.2.1. Discounted Reward Criterion

In discounted RL, the future rewards are geometrically discounted according to a discount factor γ, where $0 \leq \gamma < 1$. The performance is usually optimized in the infinite horizon [26]:

$$\lim_{N \to \infty} E^{\pi}\left(\sum_{t=0}^{N-1} \gamma^t r_t(s)\right) \tag{1}$$

The E expresses the expected value, N is the number of time steps, and $r_t(s)$ is the reward received at time t starting from state s under the policy.

2.2.2. Undiscounted (Averaged) Reward Criterion

The *averaged reward criterion* [13], which is the undiscounted RL, is where the agent selects actions maximizing its long-run average reward per step $\rho(s)$:

$$\rho^{\pi}(s) = \lim_{N \to \infty} \frac{E^{\pi}\left(\sum_{t=0}^{N-1} r_t(s)\right)}{N} \tag{2}$$

If a policy maximizes the average reward over all states, it is a *gain optimal policy*. Usually, average reward $\rho(s)$ can be denoted as ρ, which is state-independent [27], formulated as $\rho^{\pi}(x) = \rho^{\pi}(y) = \rho^{\pi}, \forall x, y \in S$ when the resulting Markov chain with policy π is ergodic (aperiodic and positive recurrent) [28].

To solve an average reward MDP problem, a stationary policy π maximizing the average reward ρ needs to be determined. To do so, the *average adjusted sum* of rewards earned following a policy π is defined as:

$$V^{\pi}(s) = E^{\pi}\left(\sum_{t=0}^{N \to \infty} (r_t - \rho^{\pi})\right) \tag{3}$$

The $V^{\pi}(s)$ can also be called a *bias* or *relative value*. Therefore, the optimal relative value for a state–action pair (s, a) can be written as:

$$V(s, a) = r^{a}(s, s') - \rho + \max_{b} V(s', b) \forall s \in S \text{ and } \forall a \in A \tag{4}$$

where $r^{a}(s, s')$ denotes the immediate reward of action a in state s when the next state is s', ρ is the average reward, and $\max_b V(s', b)$ is the maximum relative value in state s' among all possible actions b. Equation (4) is also known as the Bellman equation for an average reward MDP [28].

2.3. Integrating Reward Criterions in ACS2

Despite the ACS's *latent-learning* capabilities, the RL is realized using two classifier metrics-reward $cl.r$ and immediate reward $cl.ir$. The latter stores the immediate reward predicted to be received after acting in a particular situation and is used mainly for model exploitation where the reinforcement might be propagated internally. The reward parameter $cl.r$ stores the reward predicted to be obtained in the long run.

For the first version of ACS, Stolzmann proposed a *bucket-brigade* algorithm to update the classifier's reward r_c [20,29]. Let c_t be the active classifier at time t and c_{t+1} the active classifier at time $t + 1$:

$$r_{c_t}(t+1) = \begin{cases} (1 - b_r) \cdot r_{c_t}(t) + b_r \cdot r(t+1), & \text{if } r(t+1) \neq 0 \\ (1 - b_r) \cdot r_{c_t}(t) + b_r \cdot r_{c_{t+1}}(t), & \text{if } r(t+1) = 0 \end{cases} \tag{5}$$

where $b_r \in [0, 1]$ is the *bid-ratio*. The idea is that if there is no environmental reward at time $t + 1$, then the currently active classifier c_{t+1} gives a payment of $b_r \cdot r_{c_{t+1}}(t)$ to the previous active classifier c_t. If there is an environmental reward $r(t + 1)$, then $b_r \cdot r(t + 1)$ is given to the previous active classifier c_t.

Later, Butz adopted the Q-learning idea in ACS2 alongside other modifications [30]. For the agent to learn the optimal behavioral policy, both the reward $cl.r$ and intermediate reward $cl.ir$ are continuously updated. To assure maximal Q-value, the quality of a classifier is also considered assuming that the reward converges in common with the anticipation's accuracy. The following updates are applied to each classifier cl in action set $[A]$ during every trial:

$$\begin{aligned} cl.r &= cl.r + \beta\left(\phi(t) + \gamma \max_{cl' \in [M](t+1) \wedge cl'.E \neq \{\#\}^L} (cl'.q \cdot cl'.r) - cl.r\right) \\ cl.ir &= cl.ir + \beta(\phi(t) - cl.ir) \end{aligned} \tag{6}$$

The parameter $\beta \in [0,1]$ denotes the learning rate and $\gamma \in [0,1)$ is the discount factor. With a higher β value, the algorithm takes less care of past encountered cases. On the other hand, γ determines to what extent the reward prediction measure depends on future reward.

Thus, in the original ACS2, the calculation of the discounted reward estimation at a specific time t is described as $Q(t)$, which is part of Equation (6):

$$Q(t) \leftarrow \phi(t) + \gamma \max_{cl' \in [M](t+1) \wedge cl'.E \neq \{\#\}^L} (cl'.q \cdot cl'.r) \tag{7}$$

The modified ACS2 implementation replacing the discounted reward with the averaged version with the formula $R(t)$ is defined below (Equation (8)):

$$R(t) = \phi(t) - \rho + \max_{cl' \in [M](t+1) \wedge cl'.E \neq \{\#\}^L} (cl'.q \cdot cl'.r) \tag{8}$$

The definition above requires an estimate of the average reward ρ. Equation (4) showed that the maximization of the average reward is achieved by maximizing the relative value. The next sections will propose two variants of setting it to use the average reward criterion for internal reward distribution. The altered version is named AACS2, which stands for *Averaged ACS2*.

As the next operation in both cases, the reward parameter of all classifiers in the current action set $[A]$ is updated using the following formula:

$$cl.r \leftarrow cl.r + \beta(R - cl.r) \tag{9}$$

where β is the learning rate and R was defined in Equation (8).

2.3.1. AACS2-v1

The first variant of the AACS2 represents ρ parameter as the ratio of the total reward received along the path to reward and the average number of steps needed. It is initialized as $\rho = 0$, and its update is executed as the first operation in RL using the Widrow–Hoff delta rule (Equation (10)). The update is also restricted to be executed only when the agent chooses the action greedily during the explore phase:

$$\rho \leftarrow \rho + \zeta[\phi - \rho] \tag{10}$$

The ζ parameter denotes the learning rate for average reward and is typically set at a very low value. This ensures a nearly constant value of average reward for the update of the reward, which is necessary for the convergence of average reward RL algorithms [31].

2.3.2. AACS2-v2

The second version is based on the XCSAR proposition by Zang [12]. The only difference from the AACS2-v1 is that the estimate is also dependent on the maximum classifier fitness calculated from the previous and current match set:

$$\rho \leftarrow \rho + \zeta[\phi + \max_{cl \in [M](t) \wedge cl.E \neq \{\#\}^L} (cl.q \cdot cl.r) - \max_{cl \in [M](t+1) \wedge cl.E \neq \{\#\}^L} (cl.q \cdot cl.r) - \rho] \tag{11}$$

2.4. Testing Environments

This section will describe Markovian environments chosen for evaluating the introduction of the average reward criterion. They are sorted from simple to more advanced, and each of them has different features allowing us to examine the difference between using discounted and undiscounted reward distribution.

2.4.1. Corridor

The corridor is a 1D multi-step, linear environment introduced by Lanzi to evaluate the XCSF agent [32]. In the original version, the agent location was described by a value between $[0, 1]$. When one of the two possible actions (move left or move right) was executed, a predefined *step-size* adjusted the agent's current position. When the agent reaches the final state $s = 1.0$ the reward $\phi = 1000$ is paid out.

In this experiment, the environment is discretized into n unique states (Figure 3). The agent can still move in both directions, and a single trial ends when the terminating state is reached or the maximum number of steps is exceeded.

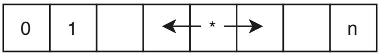


Figure 3. The Corridor environment. The agent (denoted by "*") is inserted randomly and its goal is to reach the final state n by executing two actions-moving left or right.

2.4.2. Finite State World

Barry [33] introduced the *Finite State World* (FSW) environment to investigate the limits of XCS performance in long multi-steps environments with a delayed reward. It consists of *nodes* and directed *edges* joining the nodes. Each node represents a distinct environmental state and is labeled with a unique state identifier. Each edge represents a possible transition path from one node to another and is also labeled with the action(s) that will cause the movement. An edge can also lead back to the same node. The graph layout used in the experiments is presented in Figure 4.

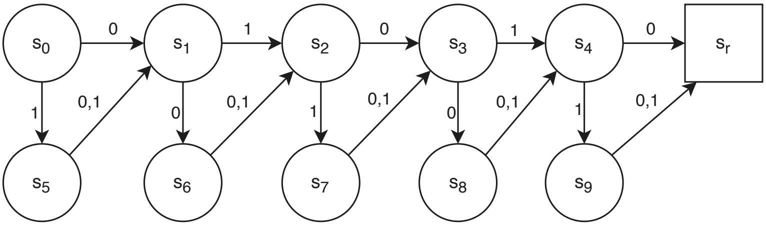


Figure 4. A Finite State World of length 5 (FSW-5) for a delayed reward experiment.

Each trial always starts in state s_0, and the agent's goal is to reach the final state s_r. After doing so, the reward $\phi = 100$ is provided, and the trial ends. The environment has a couple of interesting properties. First, it can be easily scalable, just by changing the number of nodes, which will change the action chain length. It also enables the agent to choose the optimal route at each state (where the sub-optimal ones do not prevent progress toward the reward state).

2.4.3. Woods

The Woods1 [34] environment is a two-dimensional rectilinear grid containing a single configuration of objects that is repeated indefinitely in the horizontal and vertical directions (Figure 5). It is a standard testbed for classifier systems in multi-step environments. The agent's learning task is to find the shortest path to food.

There are three types of objects available-food ("`F`"), rock ("`O`"), and empty cell ("`.`"). In each trial, the agent ("*") is placed randomly on an empty cell and can sense the environment by analyzing the eight nearest cells. Two versions of encoding are possible. Using binary encoding, each cell type is assigned two bits, so the observation vector has the length of 16 elements. On the other hand, using an encoding with the alphabet $\{0, F, .\}$, the observation vector is compacted to the length of 8.

In each trial, the agent can perform eight possible moves. When the resulting cell is empty, it is allowed to change the position. If its type is a block, then it stays in place, and one time-step elapses. The trial ends when the agent reaches the food providing the reward $\phi = 1000$.

```
.......................................................
.00F..00F..00F..00F..00F..00F..00F..00F..00F..00F..00F.
.000..000..000..000..000..000..000..000..000..000..000.
.000..000..000..000..000..000..000..000..000..000..000.
.......................................................
.......................................................
.00F..00F..00F..00F..00F..00F..00F..00F..00F..00F..00F.
.000..000..000..000..000..000..000..000..000..000..000.
.000..000..000..000..000..000*.000..000..000..000..000.
.......................................................
.......................................................
.00F..00F..00F..00F..00F..00F..00F..00F..00F..00F..00F.
.000..000..000..000..000..000..000..000..000..000..000.
.000..000..000..000..000..000..000..000..000..000..000.
.......................................................
```

Figure 5. Environment Woods1 with an animat "$*$". Empty cells are denoted by "`.`".

3. Results

The following section describes the differences observed between using the ACS2 with standard discounted reward distribution and two proposed modifications. In all cases, the experiments were performed in an explore–exploit manner, where the mode was alternating in each trial. Additionally, for better reference and benchmarking purposes, basic implementations of Q-Learning and R-Learning algorithms were also introduced and used with the same parameter settings as ACS2 and AACS2. The most important thing was to distinguish whether the new reward distribution proposition still allows the agent to successfully update the classifier's parameter to allow the exploitation of the environment. To illustrate this, figures presenting the number of steps to the final location, estimated average change during learning, and the reward payoff-landscape across all possible state–action pairs were plotted.

For the reproduction purposes, all the experiments were performed in Python language. A PyALCS (https://github.com/ParrotPrediction/pyalcs) [35] framework was used for implementing additional AACS2-v1 and AACS2-v2 agents (https://github.com/ParrotPrediction/pyalcs) and all the environments used are implemented according to the OpenAI Gym [36] in a separate repository (https://github.com/ParrotPrediction/openai-envs). Publicly available interactive Jupyter notebooks presenting all results are available for reproduction here (https://github.com/ParrotPrediction/pyalcs-experiments).

3.1. Corridor 20

The following parameters were used: $\beta = 0.8$, $\gamma = 0.95$, $\epsilon = 0.2$, $\zeta = 0.0001$. The experiments were run on 10,000 trials in total. Because there is only one state to be perceived by the agent, the genetic generalization feature was disabled. The corridor of size $n_{corridor} = 20$ was tested, but similar results were also obtained for greater sizes.

The average number of steps can be calculated $\frac{\sum_0^{n_{corridor}} n}{n_{corridor}-1}$, which for the tested environment gives the approximate value of 11.05. It is seen that the average reward per step in this environment should be close to 90.47.

Figure 6 demonstrates that the environment is learned in all cases. The anticipatory classifier systems obtained an optimal number of steps after the same number of exploit trials, which is about 200. In addition, the AACS2-v2 updates the ρ value more aggressively in earlier phases, but the estimate converges near the optimal reward per step.

For the payoff-landscape, all allowed state–action pairs were identified in the environment (38 in this case). The final population of learning classifiers was established after 100 trials and was the same size. Both Q-table and R-learning tables were filled in using the same parameters and number of trials.

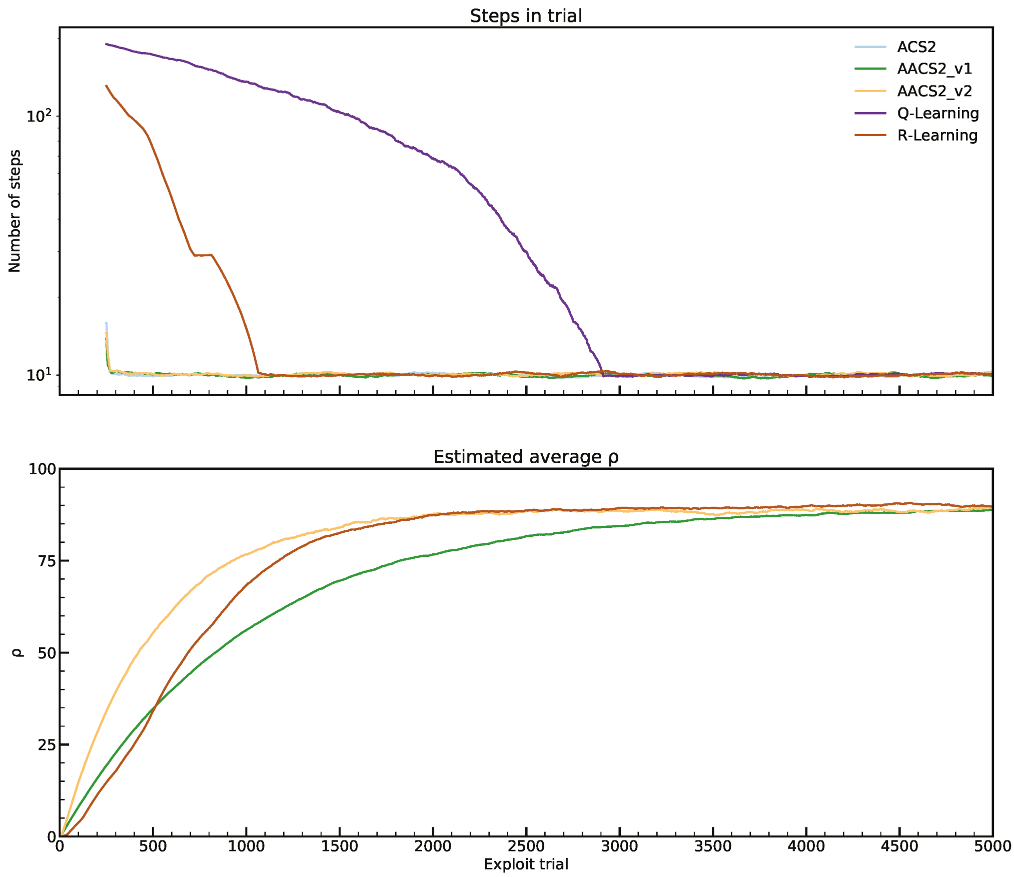


Figure 6. Performance on Corridor 20 environment. Plots are averaged in ten experiments. For the number of steps, a logarithmic scale ordinate and a moving average with window 250 was applied.

Figure 7 depicts the differences in the payoff-landscapes. The relative distance between adjacent state–action pairs can be divided into three groups. The first one relates to the discounted reward agents (ACS2, Q-Learning). Both generate almost a similar reward payoff for each state–action. Later, there is the R-Learning algorithm, which estimates the ρ value and separates states evenly. Furthermore, two AACS2 agents are performing very similarly. The ρ value calculated by the R-Learning algorithm is lower than the average estimation by the AACS2 algorithm.

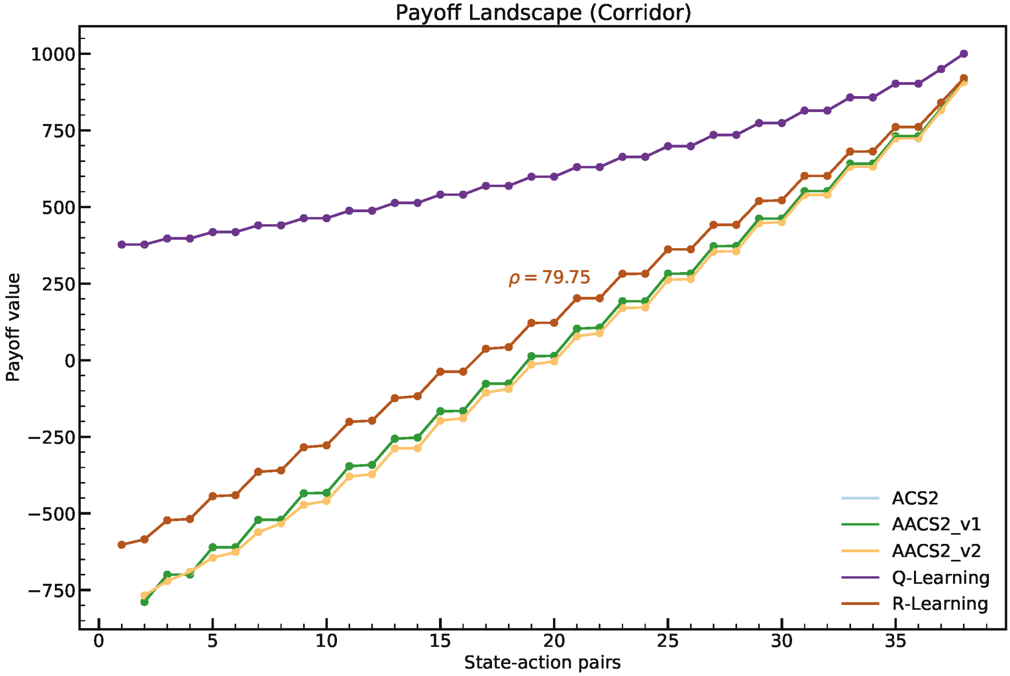


Figure 7. Payoff Landscape for Corridor 20 environment. Payoff values were obtained after 10,000 trials. For the Q-Learning and R-Learning, the same learning parameters were applied. The ACS2 and Q-Learning generate exactly the same payoffs for each state–action pair.

3.2. Finite State Worlds 20

The following parameters were selected: $\beta = 0.5$, $\gamma = 0.95$, $\epsilon = 0.1$, $\zeta = 0.0001$. The experiments were performed in 10,000 trials. Similarly as before, there is only one state observed, and the genetic generalization mechanism remains turned off. The size of the environments was chosen to be $n_{fsw} = 10$, resulting in $2n_{fsw} + 1 = 21$ distinct states.

Figure 8 presents that agents are capable of learning a more challenging environment without any problems. It takes about 250 trials to reach the reward state performing an optimal number of steps. Like in the corridor environment from Section 3.1, the ρ parameter converges with the same dynamics.

The payoff-landscape Figure 9 shows that the average value estimate is very close to the one calculated by the R-learning algorithm. The difference is mostly visible in the state–action pairs located afar from the final state. The discounted versions of the algorithms performed precisely the same.

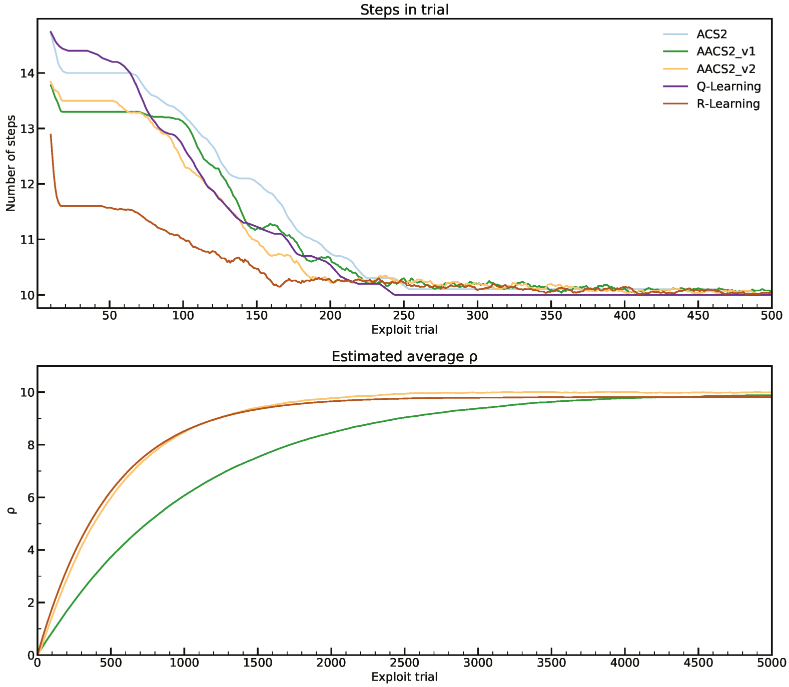


Figure 8. Performance on the FSW-10 environment.Plots are averaged in ten experiments. For the number of steps, a moving average with window 25 was applied. Notice that the abscissa on both plots is scaled differently.

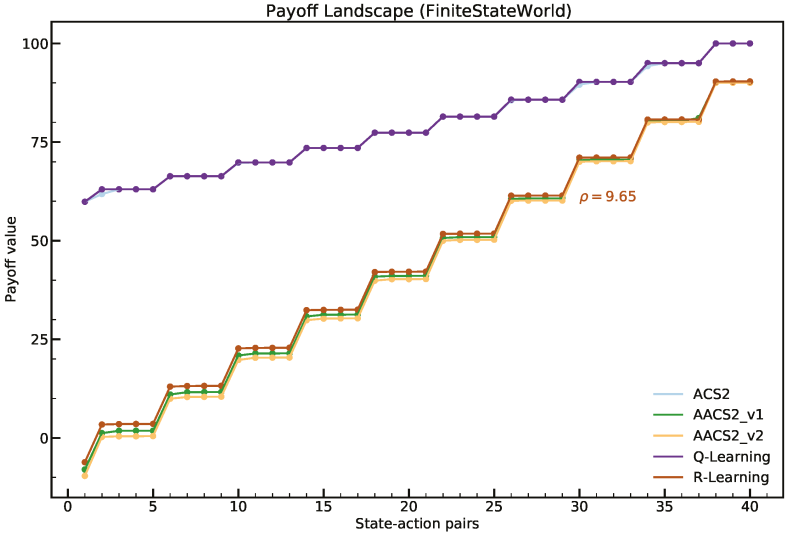


Figure 9. Payoff Landscape for the FSW-10 environment. Payoff values were obtained after 10,000 trials. For the Q-Learning and R-Learning, the same learning parameters were applied.

3.3. Woods1

The following parameters were used: $\beta = 0.8$, $\gamma = 0.95$, $\epsilon = 0.8$, $\zeta = 0.0001$. Each environmental state was encoded using three bits, so the perception vector passed to agent has the length of 24. The genetic generalization mechanism was enabled with the parameters: mutation probability $\mu = 0.3$, cross-over probability $\chi = 0.8$, genetic algorithm application threshold $\theta_{ga} = 100$. The experiments were performed in 50,000 trials and repeated five times.

The optimal number of steps in the Woods1 environment is 1.68, so the maximum average reward can be calculated as 1000/1.68, i.e., 595.24.

Figure 10 shows that the ACS2 did not manage to learn the environment successfully—the number of steps performed in the exploit trial is not stable and varies much higher than the optimal value. On the other hand, both AACS2 versions managed to function better. The AACS2-v2 stabilizes faster and with weaker fluctuations. The best performance was obtained for the Q-Learning and R-Learning algorithm that managed to learn the environment in less than 1000 trials. The average estimate ρ value converges at the value of 385 for both cases after 50,000 trials, which is still not optimal.

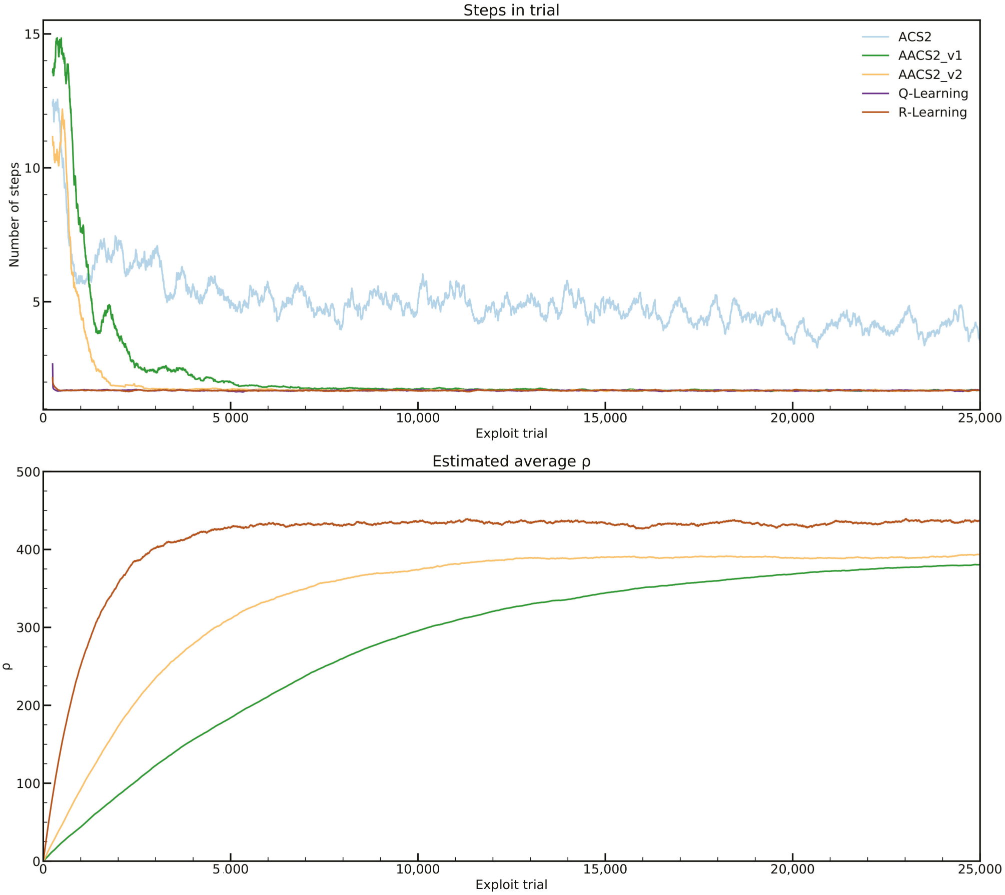


Figure 10. Performance in the Woods1 environment. For brevity, the number of steps is averaged on 250 latest exploit trials. Both AACS2 variants managed to converge to the optimal number of steps.

What is interesting is that neither ACS2 nor AACS2 population settled to the final size. Figure 11 demonstrates the difference in size for each algorithm between total population size and the number of reliable classifiers. Even though the algorithm manages to find the shortest path for AACS2, the number of created rules is greater than all unique state–action pairs in the environment, which is 101. The experiment was also performed ten times longer (one million trials) to see if the correct rules will be discovered, but that did not happen.

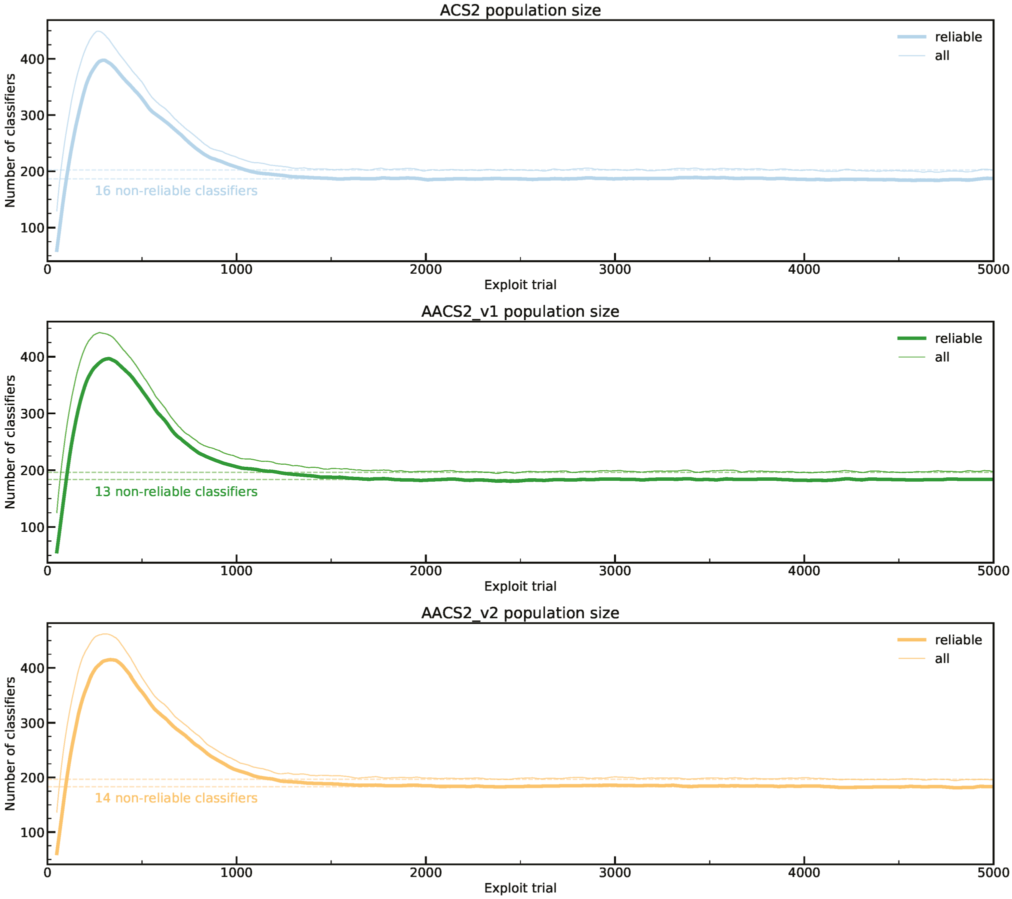


Figure 11. Comparison of classifier populations in Woods1 environment. None of the algorithms managed to create stable population size. The number of exploit trials is narrowed to the first 5000 exploit trials, and the plots are averaged on 50 latest values for clarity.

Finally, the anticipatory classifier systems' inability to solve the environment is depicted in payoff-landscape Figure 12. The Q-Learning and R-Learning have three spaced threshold levels, corresponding to states where the required number of steps to the reward states is 1, 2, and 3. All ALCS struggle to learn the correct behavior anticipation. The number of classifiers detected for each state–action is greater than optimal.

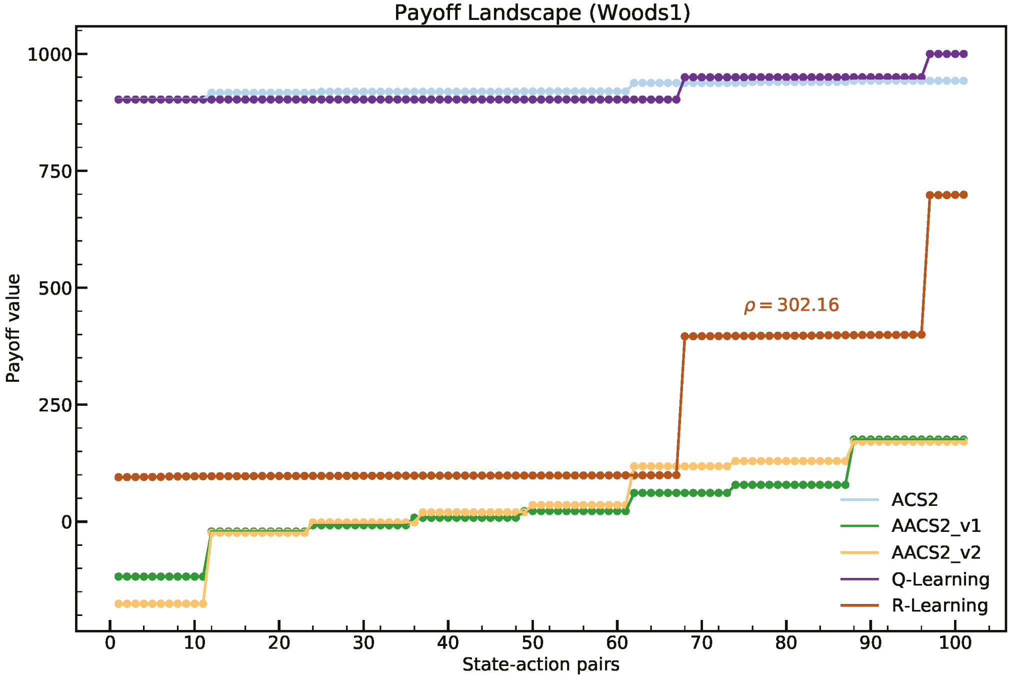


Figure 12. Payoff-landscape in the Woods1 environment. Three threshold levels are visible for the Q-Learning and R-Learning algorithms representing states in the environment with a different number of steps to the reward state.

4. Discussion

Experiments performed indicated that anticipatory classifier systems with the averaged reward criterion can be used in multi-step environments. The new system AACS2 varies only in a way the classifier reward $cl.r$ metric is calculated. The clear difference between the discounted criterion is visible on the payoff landscapes generated from the testing environments. The AACS2 can produce a distinct payoff-landscape with uniformly spaced payoff levels, which is very similar to the one generated by the R-learning algorithm. When taking a closer look, all algorithms generate step-like payoff-landscape plots, but each particular state–action pairs are more distinguishable when the reward-criterion is used. The explanation of why the agent moves toward the goal at all can be found in Equation (8)—it is able to find the next best action by using the best classifiers' fitness from the next match set.

In addition, the rate at which the average estimate value ρ converges is different for AACS2-v1 and AACS2-v2. Figures 6, 8, and 10 demonstrate that the AACS2-v2 settles to the final value faster, but also has greater fluctuations. That is caused by the fact that both match sets' maximum fitness is considered when updating the values. Zang also observed this and proposed that the learning rate ζ in Equation (11) could decay over time [12]:

$$\zeta \leftarrow \zeta - \frac{\zeta^{max} - \zeta^{min}}{NumOfTrials} \tag{12}$$

where ζ^{max} is the initial value of ζ, and ζ^{min} is the minimum learning rate required. The update should take place at the beginning of each exploration trial.

In addition, the fact that the optimal ρ value was not optimal value might be caused by the exploration strategy adopted. The selected policy was ϵ-greedy. Because the estimated average reward is updated only when the greedy action is executed, the number of greedy actions to be performed during the exploration trial is uncertain. In addition, the probability distribution when the agent observes the rewarding state might be too low in order to enable the estimated average reward to reach optimal value. This was observed during the experimentation—the ρ value was very dependent on the ϵ parameter used.

To conclude, additional research would be beneficial paying extra attention to:

- the performance on longer and more complicated environments (like containing irrelevant perception bits),
- the impact of different action selection policies, especially those used in ALCSs like the Action-Delay, Knowledge-Array or Optimistic Initial Qualities [18,37],
- fine-tuning ϵ parameter for optimal average reward estimation,
- difference between two versions of AACS2 in terms of using fitness from the match-set. The estimation is calculated differently in both cases, especially in the early phase of learning.

Author Contributions: Conceptualization, O.U. and N.K.; data curation, N.K.; formal analysis, N.K. and O.U.; funding acquisition, O.U.; investigation, N.K.; methodology, N.K. and O.U.; project administration, O.U.; resources, N.K. and O.U.; software, N.K.; supervision, O.U.; validation, O.U. and N.K., visualization, N.K.; writing—original draft preparation, N.K. and O.U.; writing—review and editing, N.K. and O.U. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AACS2	Averaged Anticipatory Classifier System 2
ACS	Anticipatory Classifier System
ALCS	Anticipatory Learning Classifier System
ALP	Anticipatory Learning Process
FSW	Finite State World
LCS	Learning Classifier System
MDP	Markov Decision Problem
RL	Reinforcement Learning

References

1. Holland, J. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In *Machine learning: An Artificial Intelligence Approach*; Morgan Kaufmann: San Francisco, CA, USA, 1986.
2. Bacardit, J.; Krasnogor, N. *BioHEL: Bioinformatics-Oriented Hierarchical Evolutionary Learning*; University of Nottingham: Nottingham, UK, 2006.
3. Urbanowicz, R.J.; Moore, J.H. ExSTraCS 2.0: Description and evaluation of a scalable learning classifier system. *Evol. Intell.* **2015**, *8*, 89–116. [CrossRef] [PubMed]
4. Borna, K.; Hoseini, S.; Aghaei, M.A.M. Customer satisfaction prediction with Michigan-style learning classifier system. *SN Appl. Sci.* **2019**, *1*, 1450. [CrossRef]
5. Liang, M.; Palado, G.; Browne, W.N. Identifying Simple Shapes to Classify the Big Picture. In Proceedings of the 2019 International Conference on Image and Vision Computing New Zealand (IVCNZ), Dunedin, New Zealand, 2–4 December 2019; pp. 1–6.
6. Tadokoro, M.; Hasegawa, S.; Tatsumi, T.; Sato, H.; Takadama, K. Knowledge Extraction from XCSR Based on Dimensionality Reduction and Deep Generative Models. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 1883–1890.
7. Pätzel, D.; Stein, A.; Nakata, M. An overview of LCS research from IWLCS 2019 to 2020. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, Cancún, Mexico, 8–12 July 2020; pp. 1782–1788.
8. Wilson, S.W. Classifier fitness based on accuracy. *Evol. Comput.* **1995**, *3*, 149–175. [CrossRef]
9. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
10. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: New York, NY, USA, 2014.

11. Tharakunnel, K.; Goldberg, D.E. XCS with Average Reward Criterion in Multi-Step Environment. 2002. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.3517 (Assessed on 1 October 2020).
12. Zang, Z.; Li, D.; Wang, J.; Xia, D. Learning classifier system with average reward reinforcement learning. *Knowl. Based Syst.* **2013**, *40*, 58–71. [CrossRef]
13. Schwartz, A. A reinforcement learning method for maximizing undiscounted rewards. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 June 1993; Volume 298, pp. 298–305.
14. Singh, S.P. Reinforcement learning algorithms for average-payoff Markovian decision processes. In *AAAI-94 Proceedings*; AAAI Press: Menlo Park, CA, USA, 1994; Volume 94, pp. 700–705.
15. Hoffmann, J.; Sebald, A. Lernmechanismen zum Erwerb verhaltenssteuernden Wissens. *Psychol. Rundsch.* **2000**, *51*, 1–9. [CrossRef]
16. Butz, M.V. *Anticipatory Learning Classifier Systems*; Springer Science & Business Media: New York, NY, USA, 2002; Volume 4.
17. Stolzmann, W. *Antizipative Classifier Systems*; Shaker: Aachen, Germany, 1997.
18. Kozlowski, N.; Unold, O. Investigating exploration techniques for ACS in discretized real-valued environments. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, Cancún, Mexico, 8–12 July 2020; pp. 1765–1773.
19. Butz, M.V.; Stolzmann, W. An Algorithmic Description of ACS2. In *Advances in Learning Classifier Systems*; Lanzi, P.L., Stolzmann, W., Wilson, S.W., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 211–229.
20. Stolzmann, W. An introduction to anticipatory classifier systems. In *International Workshop on Learning Classifier Systems*; Springer: Berlin, Germany, 1999; pp. 175–194.
21. Unold, O.; Rogula, E.; Kozłowski, N. Introducing Action Planning to the Anticipatory Classifier System ACS2. In *International Conference on Computer Recognition Systems*; Springer: Berlin, Germany, 2019; pp. 264–275.
22. Orhand, R.; Jeannin-Girardon, A.; Parrend, P.; Collet, P. PEPACS: Integrating probability-enhanced predictions to ACS2. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, Cancún, Mexico, 8–12 July 2020; pp. 1774–1781.
23. Orhand, R.; Jeannin-Girardon, A.; Parrend, P.; Collet, P. BACS: A Thorough Study of Using Behavioral Sequences in ACS2. In *International Conference on Parallel Problem Solving from Nature*; Springer: Berlin, Germany, 2020; pp. 524–538.
24. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
25. Mahadevan, S. Sensitive discount optimality: Unifying discounted and average reward reinforcement learning. In *ICML*; Citeseer: University Park, PA, USA, 1996; pp. 328–336.
26. Andrew, A.M. *Reinforcement Learning: An Introduction*; Adaptive Computation and Machine Learning Series; Sutton, R.S., Barto, A.G., Eds.; MIT Press (Bradford Book): Cambridge, MA, USA, 1998; p. 322, ISBN 0-262-19398-1.
27. Mahadevan, S. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Mach. Learn.* **1996**, *22*, 159–195. [CrossRef]
28. Puterman, M.L. Markov decision processes. In *Handbooks in Operations Research and Management Science*; Elsevier: Amsterdam, The Netherlands, 1990; Volume 2, pp. 331–434.
29. Holland, J.H. Properties of the bucket brigade. In Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, 24–26 July 1985; pp. 1–7.
30. Butz, M.V. ACS2. In *Anticipatory Learning Classifier Systems*; Springer: Berlin, Germany, 2002; pp. 23–49.
31. Gosavi, A. An algorithm for solving semi-Markov decision problems using reinforcement learning: Convergence analysis and numerical results. Ph.D. Thesis, University of South Florida, Tampa, FL, USA, 2000.
32. Lanzi, P.L.; Loiacono, D.; Wilson, S.W.; Goldberg, D.E. XCS with computed prediction in continuous multistep environments. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Scotland, UK, 2–5 September 2005; Volume 3, pp. 2032–2039.
33. Barry, A. XCS Performance and Population Structure in Multi-Step Environments. Ph.D. Thesis, Queen's University of Belfast, Belfast, UK, 2000.
34. Wilson, S.W. ZCS: A zeroth level classifier system. *Evol. Comput.* **1994**, *2*, 1–18. [CrossRef]
35. Kozlowski, N.; Unold, O. Integrating anticipatory classifier systems with OpenAI gym. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Kyoto, Japan, 15–19 July 2018; pp. 1410–1417.
36. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
37. Butz, M.V. Biasing exploration in an anticipatory learning classifier system. In *International Workshop on Learning Classifier Systems*; Springer: Berlin, Germany, 2001; pp. 3–22.

applied sciences

MDPI

Article

Gaussian Mixture Models for Detecting Sleep Apnea Events Using Single Oronasal Airflow Record

Hisham ElMoaqet [1,*], Jungyoon Kim [2,*], Dawn Tilbury [3], Satya Krishna Ramachandran [4], Mutaz Ryalat [1] and Chao-Hsien Chu [5]

[1] Department of Mechatronics Engineering, German Jordanian University, Amman 11180, Jordan; mutaz.ryalat@gju.edu.jo

[2] Department of Computer Science, Kent State University, Kent, OH 23185, USA

[3] Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA; tilbury@umich.edu

[4] Department of Anesthesiology, Beth Israel Deaconess Medical Center, Harvard Medical School, Boston, MA 02215, USA; skrama@bidmc.harvard.edu

[5] College of Information Sciences and Technology, Penn State University, College Park, PA 16802, USA; chc4@psu.edu

* Correspondence: hisham.elmoaqet@gju.edu.jo (H.E.); jkim78@kent.edu (J.K.)

Received: 9 October 2020; Accepted: 2 November 2020; Published: 6 November 2020

Abstract: Sleep apnea is a common sleep-related disorder that significantly affects the population. It is characterized by repeated breathing interruption during sleep. Such events can induce hypoxia, which is a risk factor for multiple cardiovascular and cerebrovascular diseases. Polysomnography, the gold standard, is expensive, inaccessible, uncomfortable and an expert technician is needed to score sleep-related events. To address these limitations, many previous studies have proposed and implemented automatic scoring processes based on fewer sensors and machine learning classification algorithms. However, alternative device technologies developed for both home and hospital still have limited diagnostic accuracy for detecting apnea events even though many of the previous investigational algorithms are based on multiple physiological channel inputs. In this paper, we propose a new probabilistic algorithm based on (only) oronasal respiration signal for automated detection of apnea events during sleep. The proposed model leverages AASM recommendations for characterizing apnea events with respect to dynamic changes in the local respiratory airflow baseline. Unlike classical threshold-based classification models, we use a Gaussian mixture probability model for detecting sleep apnea based on the posterior probabilities of the respective events. Our results show significant improvement in the ability to detect sleep apnea events compared to a rule-based classifier that uses the same classification features and also compared to two previously published studies for automated apnea detection using the same respiratory flow signal. We use 96 sleep patients with different apnea severity levels as reflected by their Apnea-Hypopnea Index (AHI) levels. The performance was not only analyzed over obstructive sleep apnea (OSA) but also over other types of sleep apnea events including central and mixed sleep apnea (CSA, MSA). Also the performance was comprehensively analyzed and evaluated over patients with varying disease severity conditions, where it achieved an overall performance of $TPR = 88.5\%$, $TNR = 82.5\%$, and $AUC = 86.7\%$. The proposed approach contributes a new probabilistic framework for detecting sleep apnea events using a single airflow record with an improved capability to generalize over different apnea severity conditions

Keywords: sleep apnea; airflow signal; Gaussian Mixture Models (GMM)

1. Introduction

Sleep apnea is a highly prevalent sleep disorder that can cause significant daytime sleepiness and result in many cardiovascular comorbidities [1–4]. It is characterized by repetitive significant airflow reductions during sleep causing recurrent hypoxia and sleep fragmentation [5–7]. When breathing doesn't completely stop but the volume of air going into the lungs is significantly reduced, then the respiratory event is called a hypopnea. More than 200 million patients worldwide are affected with sleep apnea [8].

Sleep apnea events are three types: Obstructive, central, and mixed [9]. Obstructive sleep apnea (OSA) is characterized by repetitive upper airway obstruction that limit airflow from going in to the lungs with the presence of continued respiratory effort. Central sleep apnea (CSA) is characterized by the loss of all respiratory effort during sleep due to a neurological disorder. Mixed sleep apnea (MSA) is combination of both obstructive and central sleep apnea symptoms.

Polysomnography (PSG), often called a sleep study, is the gold standard for detecting sleep apnea. Polysomnography records basic human body activities during sleep in an attended setting (sleep laboratory). This includes electrocardiogram (ECG) for heart, oronasal thermal airflow signal (FlowTh) and nasal pressure signal (NPRE) for respiration, electroencephalogram (EEG) for brain, electromyogram (EMG) for muscles, and oxygen level in the blood (SpO_2) [10,11]. Connecting a large number of sensors and wires to a subject in a dedicated sleep lab makes PSG uncomfortable, expensive, and unavailable to a large number of sleep patients in many parts of the world [9]. Moreover, clinicians need an offline inspection of the recordings to score apnea and derive the apnea-hypopnea index (AHI), which is the parameter used to establish sleep apnea and its severity [12]. Thus, the analysis process is labor-intensive and time-consuming, leading to a delayed diagnostic process and increased patient waiting lists [13–16] as well as being highly susceptible to human errors [17].

To overcome limitations of PSG, several studies have been proposed for automated detection of sleep apnea using a limited subset of signals among those involved in PSG [18]. This includes respiratory signals [19–29], ECG [16,30–32], SpO_2 [33–35], tracheal sound signals [36], or some combinations of signals listed above [37]. A number of portable devices for sleep apnea monitoring and diagnosis have been developed and are available. LifeShirt, SleepStrip, and ApneaLink are among the most popular products [38].

Respiratory airflow signal is a straightforward choice to look for simpler alternatives to PSG, since apneas are primarily defined on the basis of its amplitude oscillations [12]. According to the American Academy of Sleep Medicine (AASM), the primary sensor for identifying apnea in sleep diagnostic studies is the oronasal thermal airflow sensor [11]. Thus, several studies focused on automated detection of sleep apena events based exclusively on the analysis of this signal. In these studies, the airflow signal is analyzed in different analytic domains (linear, nonlinear, time and frequency domains) to extract features which are then used in a rule-based threshold classifier or in a "black box" machine learning model. Rule-based threshold classification has been used in [24,39–42]. Support Vector Machines (SVM) [43,44], Artificial Neural Networks (ANN) [20,45–47], linear discriminant analysis (LDA), and regression trees (CART) with the AdaBoost (AB) [22] are among the most popular machine learning models that used respiratory flow signal.

Despite their popularity in sleep apnea problems, a major limitation in classical rule-based threshold detectors is that they provide classification based on simple comparison for the features against experimentally derived thresholds while overlooking the statistical distributions for the input features as well as the output classes. Even more complex discriminative (black box) methods are based on learning a function that maps the features directly into decisions. There hasn't been much research considering probabilistic view of classification for sleep apnea detection.

Gaussian Mixture Model (GMM) is a probabilistic machine learning framework that aims at providing a richer class of density models than single Gaussian using a finite weighted mixture of Gaussian densities. It is well known as a rich framework capable of characterizing any continuous density. This framework has also shown promising results in classification problems including noisy features [48]. Nevertheless, it hasn't been well evaluated for sleep apnea detection problems.

The contribution of this paper is two fold. First, we develop a probability based classification approach for automated detection of sleep events using single oronasal airflow record. Second, we study the performance of the proposed approach over a large data set of 96 patients of different sleep apnea severity levels. Finally, we conduct a comprehensive evaluation and comparison of the proposed probabilistic framework against a rule-based classifier for the same input features as well as two previously published algorithms for apnea detection using airflow signal.

This paper is organized as follows. Section 2 describes the data set, the proposed algorithm, classification methods, and evaluation metrics. Section 3 presents results for the proposed algorithm along with a detailed comparison with related works. Section 4 discusses the results and lessons learned and 5 summarizes conclusion of the paper.

2. Materials and Methods

2.1. Data Set

The Institutional Review Board (IRB) at the University of Michigan approved this study (IRB#HUM00069035). Full polysomnography (PSG) data was collected for 96 patients at the University of Michigan Sleep Disorders Center. For each patient, polysomnography consisted of electroencephalography (EEG), electrooculography (EOG), submental and tibial electromyography (EMG), electrocardiography (ECG), two piezoelectric belts for recording plethysmography (PPG), oronasal airflow sensor (FlowTH), air pressure transducer (NPRE), digital micro-phone and pulse oximeter.

The oronasal airflow sensor used in this study is a thermocouple-based one from Respironics Model: Pro-Tech-*P*1273 (Philips Healthcare, Eindhoven, The Netherlands). Clinical annotations for respiratory events were carried out by expert clinicians from the Sleep Disorders Center at the University of Michigan (Ann Arbor, MI) and according to recommendations of the AASM [11]. Apneic events in the data set are either obstructive (OSA), central (CSA), or mixed (MSA). The data set spans different apnea severity levels as reflected by the apnea-hypopnea index (AHI) computed over night for patients in the study. 10 are non/minimal sleep apnea patients (AHI $<$ 5), 36 are mild sleep apnea patients ($5 \leq$ AHI < 15), 27 are moderate sleep apnea patients and 23 are severe sleep apnea patients (AHI $\geq$ 30). Table 1 provides distribution for the numbers and types of the apneic events per each class of the patient severity levels.

The oronasal airflow signals of 66 patients (with different apnea severity levels) were used for training the proposed modeling framework. The developed framework was then tested on 30 patients (distinct from the training data) composed of 5 none/minimal apnea , 5 mild, 5 moderate, and 15 severe sleep apnea patients.

Table 1. Distribution of Number and Types of Apneic Events per each Class of the Severity Levels.

Severity Level	OSA	CSA	MSA	Total
None/Minimal	20	312	0	332
Mild	1302	1887	78	3267
Moderate	3056	4240	400	7696
Severe	10,922	23,960	2143	37,025
Total	15,300	30,399	2621	48,320

2.2. A Data-Driven Approach for Characterizing Changes in Respiratory Baseline

According to the AASM, an apnea event is scored if there is a drop in peak thermal airflow signal excursions by $\geq$90% of the corresponding baseline for a duration $\geq$10 s [11]. Nevertheless, the airflow baseline is not precisely defined neither in the AASM Scoring Manual nor in sleep literature.

To overcome this limitation, a data driven approach will be used to derive the respiratory flow baseline from the airflow signal (FlowTH). The derived baseline will then be used to characterize dynamic changes in respiration with respect to this (dynamic) baseline in order to detect the occurrence of apneic events. To establish respiratory baseline, we will consider two important respiratory features: Inter-breath intervals and breath amplitudes.

A sliding window method will be used for detecting apnea events in the oronasal airflow signal (FlowTH). At time step t, two windows will be established. The first window (baseline window-W_b) will be used to derive the local respiratory baseline. The second window (detection window-W_m) will be used to detect apneic events based on relative changes in inter-breath intervals and breath amplitudes in W_m with respect to those in the W_b. In this study, we considered a W_b of length $L_b = 600$ s that contains the airflow measurements up to time t and a W_m of length $L_m = 100$ s that contains airflow measurements starting from time step $t+1$.

After constructing W_b and W_m, peaks and valleys of the respiratory airflow signal are detected in both windows. An example of W_b and W_m that both include an apneic event along with peak and valley detections is illustrated in Figure 1a. The inter-breath intervals and the breath amplitudes can now be extracted from W_b and W_m as follows:

$$PP_i = t_{i+1} - t_i \tag{1}$$

$$PV_i = P_i - V_i \tag{2}$$

where the airflow breath i has a peak P_i that occurs at time instance t_i, a valley V_i, an inter-breath interval PP_i, and a breath amplitude PV_i. These Equations generate sequences of inter-breath intervals and breath amplitudes in W_b and W_m as illustrated in Figure 1b,c,f,g.

After getting these sequences, it is required to extract the (inter-breath) intervals and (breath) amplitudes in W_b that contribute most to the respiratory baseline estimate of this window. Similarly, it is required to extract the intervals and amplitudes in W_m that belong to the apneic event to be detected. This can be effectively done by sorting sequences of intervals and amplitudes in both W_b and W_m based on corresponding values. For convenience, PP_i and PV_i in W_b are sorted in a descending order while those in W_m are sorted in an ascending order as illustrated in Figure 1d,e,h,i. This process will generate ordered sequences $PP_b = \{PP_i^d\}$, $PV_b = \{PV_i^d\}$, $PP_m = \{PP_i^a\}$, and $PV_m = \{PV_i^a\}$ where subscripts b and m specify W_b and W_m respectively, and superscripts a and d specify ascending and descending orders respectively.

Although the length of W_b and W_m are fixed, the number of airflow breaths in these windows typically vary during different sleep stages and across different patients. Thus, the ordered sequences (PP_b, PV_b, PP_m, and PV_m) will be filtered to keep only the intervals and amplitudes that contribute most to the baseline estimate in W_b and the apneic events in W_m respectively. This can be mathematically expressed as follows:

$$L_{PP_b} = \lfloor F_{PP_b} N_{PP_b} \rfloor \tag{3}$$

$$L_{PV_b} = \lfloor F_{PV_b} N_{PV_b} \rfloor \tag{4}$$

$$L_{PP_m} = \lfloor F_{PP_m} N_{PP_m} \rfloor \tag{5}$$

$$L_{PV_m} = \lfloor F_{PV_m} N_{PV_m} \rfloor \tag{6}$$

where F_s is the cut-off filter applied to the ordered sequence s of length N_s in order to generate a filtered sequence of length L_s where $s \in \{PP_b, PP_m, PV_b, PV_m\}$. Accordingly, the filtered sequences include the highest L_{PP_b} inter-breath intervals and L_{PV_b} breath amplitudes in W_b and the lowest L_{PP_m} intervals and L_{PV_m} amplitudes in W_m. The filter values were defined individually for each of the sequences to allow them to be tuned separately to maximize the ability to detect apneic windows. The mathematical means of the filtered sequences can now expressed as follows:

$$B_{PP_b} = \frac{1}{L_{PP_b}} \sum_{i=1}^{L_{PP_b}} PP_i^d \tag{7}$$

$$B_{PV_b} = \frac{1}{L_{PV_b}} \sum_{i=1}^{L_{PV_b}} PV_i^d \tag{8}$$

$$B_{PP_m} = \frac{1}{L_{PP_m}} \sum_{i=1}^{L_{PP_m}} PP_i^a \tag{9}$$

$$B_{PV_m} = \frac{1}{L_{PV_m}} \sum_{i=1}^{L_{PV_m}} PV_i^a, \tag{10}$$

where B_s is the mathematical mean of the filtered sequence s. The relative changes in the inter-breath intervals (I_c) and the amplitude of the breaths (A_c), with respect to the respiratory baseline, can now be computed as follows:

$$I_c = \frac{B_{PP_b} - B_{PP_m}}{B_{PP_b}} \tag{11}$$

$$A_c = \frac{B_{PV_b} - B_{PV_m}}{B_{PV_b}} \tag{12}$$

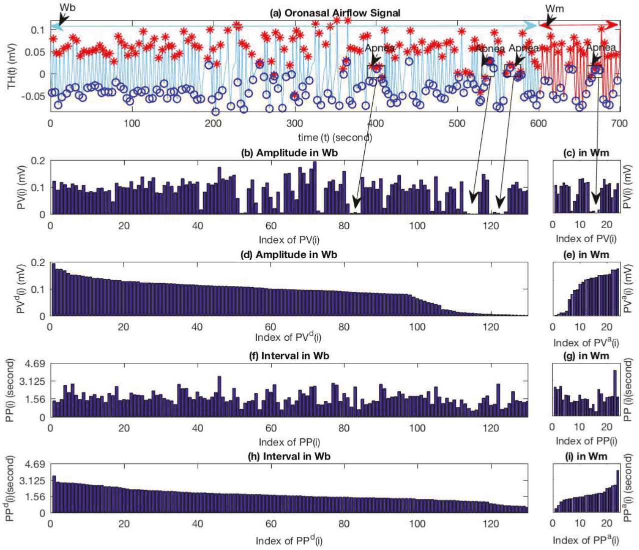


Figure 1. (**a**) Oronasal respiration signal with peak/ valley detections illustrated by (red '*')/ (blue 'o') respectively. (**b**) Sequence of extracted breath amplitudes of W_b. (**c**) Sequence of extracted breath amplitudes of W_m. (**d**) PV_b Sequence of descending ordered amplitudes in W_b. (**e**) PV_m Sequence of ascending ordered amplitudes of W_m. (**f**) Extracted inter-breath intervals of W_b. (**g**) Extracted inter-breath intervals of W_m. (**h**) PP_b sequence of descending ordered inter-breath intervals of W_b. (**i**) PP_m sequence of ascending ordered intervals of W_m. Annotations in subplot (**a**) illustrate example of apnea events present in W_b, W_m. Arrows point to the breath amplitudes from apnea events shown in W_b, W_m.

2.3. Detection of Apnea Events based on Relative Changes in Respiratory Baseline

For the classification part, we propose a probabilistic view of classification for automated detection of apnea events. We leverage a Gaussian Mixture Model (GMM) to derive a decision boundary based on probabilistic assumptions about the underlying distribution of the respiratory input features. We denote this modeling scheme as a Gaussian Mixture Model (GMM) classifier. In order to demonstrate the improvement achieved by considering GMM as a generative machine learning model, we compare results with a classical threshold-based detector that uses the same input features for automated detection of apnea events.

To prepare data for classification, a sliding window that is being successively updated each 20 s (step-size = 20 s) was applied over the oronasal airflow signal. At each of the steps, W_b and W_m are constructed. Then, Equations (1)–(12) are applied to compute I_c and A_c (input features to the classification model) while W_m provides classification label based on whether or not an apnea event was clinically scored in this window. Considering our data set with 66 patients for training and 30 for testing, Table 2 shows the distribution of the data segments and corresponding labels for both training and test sets.

Table 2. Segments for training and testing classification models.

Labels/ Segments	Training Set	Testing Set	Total
Normal	208,456	20,789	229,245
Apnea	81,055	8048	89,103
Total	289,511	28,837	318,348

2.3.1. Rule-Based Threshold Based Classification

The rule-based classifier detects apnea in detection windows (W_m) when the input features I_c and A_c both activate the classification rules $\hat{I}_{lb} \leq I \leq \hat{I}_{ub}$ and $A_c > \hat{A}$ where $\hat{I}_{lb}$, $\hat{I}_{ub}$ are the classification thresholds for I_c and $\hat{A}$ is the classification threshold for A_c. An exhaustive search approach is applied for each of these thresholds in order to learn their optimal values. A novel approach was used to fit the rule-based classifier and learn the classification rules using a two step optimization method. The classification thresholds for I_c are optimized first to obtain the receiver operating characteristics curve (ROC) with the maximum area under ROC (AUC) over our training data. Once the I_c classification rule is learned, the optimal classification threshold for A_c is learned by searching along the selected receiver operating curve (with maximum AUC) for the threshold that provides the maximum sensitivity (TPR) that constrains (FPR) not to exceed the maximum acceptable limit of 20% ($FPR \leq 20\%$). More details about the derivation and tuning of the rule-based classifier can be found in our recent study [49].

2.3.2. Classification with Gaussian Mixture Models (GMM)

A Gaussian mixture model (GMM) is a probabilistic modeling framework. In this model, the probability density function (PDF) of $\mathbf{x} \in R^d$ is defined as a finite weighted sum of k Gaussian distributions:

$$p(\mathbf{x}|\boldsymbol{\Theta}) = \sum_{i=1}^{k} \gamma_i p(\mathbf{x}|\theta_m) \tag{13}$$

such that $\mathbf{x}$ is the 2-dimensional feature vector $[I_c, A_c]^T$ computed every time W_b and W_m are constructed, $\sum_{i=1}^{k} \gamma_i = 1$, $\boldsymbol{\Theta}$ is the mixture model, γ_i corresponds to the weight of component i, and the density of each component is given by the normal probability distribution:

$$p(\mathbf{x}|\theta_m) = \frac{|\boldsymbol{\Sigma}_m|^{-\frac{1}{2}}}{(2\pi)^{d/2}} \exp\bigg\{ -\frac{1}{2}(\mathbf{x}-\mu_m)^T \boldsymbol{\Sigma}_m{}^{-1}(\mathbf{x}-\mu_m)\bigg\} \tag{14}$$

The parameters γ, the mean μ, and the covariance $\boldsymbol{\Sigma}$ are optimized during the training process using the expectation maximization algorithm [50] such that the log-likelihood of the model is maximized. During testing, a likelihood estimate is obtained for the apnea class, defined by the model $\boldsymbol{\Theta}_A$, and for the non-apneic (normal respiration) class, defined by the model $\boldsymbol{\Theta}_N$. Using the Bayesian classification formula, the likelihood estimates are combined to compute the posterior probability of apnea for the sample $\mathbf{x}$:

$$P(A|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\Theta}_A)P(A)}{p(\mathbf{x}|\boldsymbol{\Theta}_A)P(A) + p(\mathbf{x}|\boldsymbol{\Theta}_N)P(N)} \tag{15}$$

where $P(A)$ and $P(N)$ are the prior probabilities of the apnea and non-apnea (normal) classes respectively. These probabilities were set by symmetry to be equal $P(A) = P(N) = 0.5$ assuming we have no prior knowledge about them. The combination of the two GMMs and the Bayesian classification formula in Equation (15) form the GMM classifier [51].

2.4. Evaluation of Apnea Detection Results

2.4.1. Classification Performance over Detection Windows

In this paper, five statistical metrics of accuracy (ACC), true positive rate (TPR), true negative rate (TNR), positive predictive value (PPV), and F_1 score are applied to assess the performance of the proposed modeling framework over all detection windows (W_m):

$$ACC = \frac{\sum TP + \sum TN}{\sum TP + \sum FP + \sum FN + \sum TN} \times 100\% \tag{16}$$

$$TPR = \frac{\sum TP}{\sum TP + \sum FN} \times 100\% \tag{17}$$

$$TNR = \frac{\sum TN}{\sum FP + \sum TN} \times 100\% \tag{18}$$

$$PPV = \frac{\sum TP}{\sum TP + \sum FP} \times 100\% \tag{19}$$

$$F_1 = 2\frac{TPR.PPV}{TPR + PPV} \times 100\% \tag{20}$$

where TP (true positive) is the number of apneic windows that were correctly classified as such, TN (true negative) is the number of normal windows that were correctly classified as such, FP (false positive) is the number of normal windows that were falsely classified as apneic, FN (false negative) is the number of apneic windows that were missed by the classifier. ACC is a classical measure for binary classification but is not enough in this problem due to class imbalance between the apnea and normal classes [52–54]. Thus, TPR, TNR, and PPV are used to report a more detailed performance in detecting apneic and normal windows. The F_1 score considers TP and FP detections simultaneously and thus accounts to the TPR/PPV tradeoff reporting a more comprehensive idea on the overall performance of the proposed model.

2.4.2. Receiver Operating Characteristics (ROC) Curve

The Receiver Operating Characteristics (ROC) curve is an effective tool used to graphically illustrate the diagnostic ability of a binary classification system as its classification threshold is varied [55]. This curve simply plots the TPR against the false positive rate ($FPR = 100\% - TNR$) at various discrimination threshold settings. The Area Under receiver operating characteristics Curve (AUC) is used as a measure of the overall ability of the classification model to automatically detect sleep apnea events. A greater AUC indicates a more useful and effective classification model. Additionally, the ROC curve can be used for optimizing classification models by finding the operating threshold that provides the highest TPR for the allowable FPR level. This approach was used in learning the classification rules of the rule-based classifier.

3. Results

For the proposed GMM classifier (AICPV with GMM) and the rule-based threshold classifier (AICPV with Threshold), we used the PSGs of 66 patients for training, tuning, and optimizing the classifiers. The trained classifiers were then tested over the PSGs of the other distinct 30 patients.

3.1. Rule-Based Threshold Classifier

The optimal filter values were set to $F_{PP_b} = 0.3$, $F_{PV_b} = 0.4$, $F_{PP_m} = 0.1$, and $F_{PV_m} = 0.3$. The classification rules for detecting apneic windows were identified as follows

$$0.05 \leq I_c \leq 0.95 \tag{21}$$

$$0.957 \leq A_c \tag{22}$$

such that an apneic window is detected by the rule-based classifier whenever both rules are active.

3.2. GMM Classifier

Cross validation over the training data set was used for investigating and selecting the choices of parameters for the GMM models. These parameters include the number of Gaussian distributions needed to model each class, and the type of covariance matrix used (diagonal or full symmetrical). Our results show that 12 Gaussian components are needed to model the GMM of the apneic class and 11 Gaussian components are needed to model the GMM of the normal class along with diagonal covariance matrices for both GMMs.

3.3. Classification Performance Comparison over the Testing Data Set

We performed an overall evaluation for the proposed model (AICPV with GMM—AICPVwGMM) over the 30 patient test data and we compared the performance results with the rule-based classification model that uses the same input features (AICPV with Threshold - AICPVwTH). Also, we considered performance comparison with two well known published algorithms for automated apnea detection using the oronasal thermal airflow signal and similar time-domain based features [39,45]. The first algorithm implements a classical threshold based classification model [39] while the other one uses an artificial neural network classification model [45]. Note that [39] includes an additional module for classifying the type of detected apnea using a neural network classifier trained on the thoracic effort signal. Nevertheless, we just included the apnea detection module from this study since the proposed algorithm uses only a single channel of oronasal airflow.

In order to do a fair comparison, the four algorithms AICPVwGMM, AICPVwTH, Refs. [39,45] were all trained, evaluated, and tested on identical data within our data set. Table 3 comprehensively compares classification performance over the test data between these four algorithms. First, it can be noticed that the AICPVwGMM and the AICPVwTH outperform the two previously published algorithms in all performance measures of this paper. The AICPV algorithms demonstrate a higher ability to detect apnea events (reflected by their TPR) as well a higher ability to detect normal respiration patterns (reflected by their TNR) as opposed to [39,45]. An overall better classification performance of the AICPV algorithms can be demonstrated with the F_1 and AUC values compared to [39,45]. The improvement achieved with AICPV algorithms is mainly caused by the dynamic approach considered in these algorithms such that apneic events are characterized based on relative changes in airflow breath amplitudes and inter-breath intervals with respect to local respiration baseline.

Table 3. Comparison of performance over a 30-patient test data set between the rule-based threshold classifier (AICPVwTH), the proposed Gaussian Mixture Models (GMM) classifier (AICPVwGMM), and two previous related studies.

Algorithm	TPR (%)	TNR (%)	PPV (%)	ACC (%)	F_1 (%)	AUC (%)
AICPVwTH	88.3	79.4	28.2	80.2	42.7	83.9
AICPVwGMM	88.5	82.5	46.6	83.4	61.1	86.7
[45]	62.0	76.9	19.3	77.6	29.4	75.7
[39]	60.3	79.5	9.2	78.9	16.0	69.9

Comparing the performance obtained with the proposed GMM based classifier (AICPVwGMM) and the rule-based classifier (AICPVwTH), we can notice a 65.2% increase in the PPV of the apnea detections obtained with the GMM based classifier as opposed to the rule-based threshold classifier. Recognizing TPR/PPV tradeoff, and that TPR and PPV are performance metrics of competing natures, it can be noticed that using a GMM-based classifier, we can achieve a high TPR with a significantly improved PPV compared to the rule-based classifier. This can be also noticed by observing the significant increase in the F_1 score for the detections with the proposed GMM model as opposed to the rule-based one. Also, higher TNR and ACC are obtained with the proposed algorithm reflecting a higher ability to detect normal respiratory patterns. The overall classification performance indicated by AUC also shows an improved detection with the proposed GMM modeling framework.

To provide an in-depth analysis and understand the sources of improvement with the proposed model as opposed to the rule-based classification model, we did a comprehensive analysis for the test performance of the proposed algorithms over different apnea types and different apnea severity conditions. Tables 4 and 5 provide a detailed comparison between the GMM-based classifier and the rule based classifier over different apnea types and different apnea severity levels. A clear increase in the ability to detect OSA and CSA events can be noticed in the TPR of these detections using the proposed algorithm along with a significant increase in the PPV of all types of apnea detections. It can be also noticed that the TPR of the MSA detections is superior with both algorithms and that it didn't change between them which is mainly due to the fact the MSA events are minority compared to OSA and CSA events.

Looking at the detailed performance of the proposed GMM model and the rule-based classifier over different apnea severity conditions, we can notice that the best performance for the rule-based threshold classifier is in severe apnea patients and that the performance significantly degrades over less severe cases. On other hand, the GMM based classifier maintains a high ability to detect apnea events in severe patients, but more importantly, it has a significantly higher ability to detect apneic events in less severe apnea patients which can be clearly seen through the TPR of these detections. Moreover, looking at the AUC values, we can also notice an excellent overall ability to detect apnea events in severe patients for the GMM modeling framework as well as a significantly improved overall ability to detect apnea in less severe patients compared to the rule-based classification model.

Table 4. Test performance of AICPV with GMM classifier over different types of apnea and different apnea severities.

AICPVwGMM		TPR (%)	TNR (%)	PPV (%)	ACC (%)	F_1 (%)	AUC (%)
Type of Apnea	MSA	97.2	80.6	9.9	80.9	18.0	88.9
	CSA	88.7	80.6	41.8	81.7	56.8	84.6
	OSA	92.7	80.6	25.3	81.4	39.8	86.7
AHI Severity	Severe	96.4	77.3	68.7	83.8	80.3	86.7
	Moderate	98.1	77.5	32.6	79.5	48.9	87.8
	Mild	100.0	84.8	12.2	85.2	21.7	92.4
	None/Minimal	27.3	71.8	0.0	83.4	0.0	86.7
All		88.5	82.5	46.6	83.4	42.7	86.7

Table 5. Test performance of AICPV with threshold classifier over different types of apnea and different apnea severities.

AICPVwTH		*TPR* (%)	*TNR* (%)	*PPV* (%)	*ACC* (%)	F_1 (%)	*AUC* (%)
Type of Apnea	MSA	97.5	81.1	7.8	81.4	14.4	89.3
	CSA	86.8	81.1	17.1	81.4	28.6	83.4
	OSA	88.4	81.1	20.6	81.6	33.4	85.6
AHI Severity	Severe	92.4	90.3	82.5	91.0	87.2	91.4
	Moderate	79.5	80.7	32.8	80.6	46.4	80.1
	Mild	76.6	76.1	3.7	76.1	7.1	76.3
	None/Minimal	6.3	97.0	2.0	96.1	3.0	51.7
All		88.4	79.4	28.2	80.2	61.1	83.4

Finally, we performed a comprehensive analysis for the performance per class of apnea types among different patient severity levels. Table 6 evaluates how the proposed model performs on detecting different apnea types in each of apnea severity classes. As it can be seen in the table, the proposed model AIPCVwGMM maintains a high ability to detect different apnea types regardless disease severity in the test patients. MSA events are very rare in mild patients which caused low and skewed detection rates for these patients. Also, the final row in Table 6 was left empty since there are no MSA events in the class of none/minimal apnea patients. In general, the class of none/mnimal apnea reflects patients that are healthy or with few significant apnea events and so it a class of less interest compared to other disease states. Nevertheless, we kept performance results on all disease severity classes to report comprehensive assessment of the modeling framework.

It is worthy to be mentioned that the implemented algorithms were trained and tested using full overnight PSG records. Our goal is to test the proposed framework and to compare it with existing works in a more practical setting as opposed to many previous studies that considered shorter records avoiding the class imbalance problem and excluding segments with low signal to noise ratio (SNR). False positives were affected by the class imbalance problem, segments of low airflow signal quality, and irregular breathing patterns and artifacts. High airflow peak amplitudes resulting from increased respiratory effort after the end of apnea events may affect false positives by contributing falsely to the respiratory baseline. Future work may consider adding more advanced signal filtration algorithms that can allow more accurate detection incase of artifacts as well as to reject airflow segments with very low SNR.

Table 6. Detailed test performance of AICPV with GMM classifier (AICPVwGMM) over different Apnea–Hypopnea Index (AHI) severities per apnea class.

Type	AHI	*TPR* (%)	*TNR* (%)	*PPV* (%)	*ACC* (%)	F_1 (%)	*AUC* (%)
OSA	All	92.7	80.6	25.3	81.4	39.8	86.7
	Severe	97.2	77.5	59.8	82.5	74.0	87.4
	Moderate	92.8	84.8	31.3	85.4	46.8	88.8
	Mild	87.6	71.8	1.9	71.9	3.7	79.7
	Normal	0.0	93.3	0.0	93.2	NA	46.6
CSA	All	88.6	80.6	41.8	81.7	56.8	84.6
	Severe	95.6	77.5	52.8	81.3	68.0	86.6
	Moderate	85.9	84.8	23.9	84.9	37.4	85.4
	Mild	90.1	71.8	1.7	71.9	3.3	80.9
	Normal	7.6	93.3	1.1	92.4	1.9	50.4
MSA	All	97.2	80.6	9.9	80.9	18.0	88.8
	Severe	98.1	77.5	32.6	79.5	48.9	87.8
	Moderate	100	84.8	12.2	85.2	21.7	92.4
	Mild	27.3	71.8	0.1	71.8	0.0	49.5
	Normal	-	-	-	-	-	-

4. Discussion of Results

A probabilistic-based framework was developed in this study for automated apnea detection using single channel data from oronasal airflow record (FlowTH). The proposed framework leverages AASM recommendations to define apnea based on relative changes with respect to respiratory airflow baseline. to overcome the absence of a precise mathematical definition for airflow baseline, a data-driven method is developed to represent it based on two features: The breath amplitudes and inter-breath intervals. The apnea is then characterized based on relative changes in these features between two sliding windows: The baseline window which represents the current respiratory baseline and the detection window where an apneic event is to be detected.

For automatic detection of apneic events, we considered classification based on a probabilistic view using a GMM-based modeling framework. The proposed framework showed a significantly improved performance in detecting apnea compared to a rule-based classifier that uses the same input features as well as two previously published algorithms that respectively use threshold-based classification and neural networks, applied on time domain features from the same respiratory signal. Using relative changes in respiratory features to define apnea enabled a dynamic approach that accounts for continuous changes in the respiratory baseline making AICPVwTH and AICPVwGMM algorithms significantly more capable to detect apneic events than previous studies that considered absolute changes in classification features overlooking relative ones. Comparing the proposed model AICPVwGMM with AICPVwTH that uses a rule-based classifier showed a significantly improved performance for the GMM model characterized by achieving high TPR with a significantly improved overall PPV for all types of apnea detections. The proposed model also allows much better performance over different apnea severity levels compared to the rule-based classifier which showed best performance over severe apnea patients only with significantly degraded performance over less severe disease classes.

In recent years, many studies considered oronasal thermal airflow signal for automated apnea detection [19–21,39,42–45,56]. Nevertheless, patients with severe and moderate OSA conditions have been a major focus of many of these studies while not giving sufficient attention to less severe patient populations or other types of apnea events. The present results highlight the importance of evaluating apnea models on patients of varying severity conditions as well as on different apnea types. This also agrees with previous literature which demonstrated that high performance accuracies achieved with patients with high severity levels may not be generalizable to other groups of patients [57,58]. The detailed analysis presented in this paper using patients of varying apnea severity conditions and different apnea types provides the basis for a more comprehensive understanding of the performance of apnea detection systems.

Comparative analysis between the performance of proposed modeling framework and previously published research highlights some of the previously reported limitations for single-respiration channel based apnea detection methods. In particular, previous studies have reported significant fall in the diagnostic accuracy of automated sleep systems that measure two or fewer physiological parameters as opposed to those that measure three or more physiologic variables [38,57]. Although many automated sleep systems have proved effectiveness assisting lab PSGs and at home, they still cannot completely replace dedicated centers for sleep studies [59].

The present study leverages AASM recommendations for apnea detection in many aspects. We considered the AASM recommended sensor for scoring apneic events in PSG diagnostic studies which is the oronasal thermal airflow sensor. The criteria in the AASM manual were also employed for scoring an event after a drop in peak thermal airflow signal excursions by $\geq$ 90% of the corresponding baseline for a duration $\geq$10 s [11]. Nevertheless, there are many sources of uncertainty in the criteria defined in literature. First, the AASM doesn't provide a precise definition for the respiratory baseline. Second, the published criteria for mathematical scoring apnea are not consistent and vary with different standards [60]. Moreover, the high intraobserver and interobserver variability due to human scoring and human errors [17] significantly affect the robustness and performance of automated sleep systems.

Adopting a probabilistic framework in the proposed study provides an efficient approach to propagate different sources of uncertainty using a data-driven modeling framework optimized with respect to the ability to detect apnea events. Interestingly, using the proposed GMM-based classification system showed an overall improved performance in apnea detection and a more consistent and generalizable performance across patients with different severity levels.

Finally, the presented approach focuses on automated detection for apneic events using oronasal airflow respiration signal. Future work many extend the algorithm by adding an additional input through the nasal pressure signal in order to study dynamical changes in this signal during hypopnea as recommended by the AASM. Using the oronasal airflow and the nasal pressure signals will enable detecting both apnea and hypopnea events needed to estimate the AHI in order to provide a diagnosis for sleep apnea severity. Additionally, this will improve the ability to estimate respiratory baseline by incorporating two signals instead of one leading potentially to a decreased number of false positive detections. Moreover, future work may consider adding input from the respiratory effort signal to provide diagnosis for the type of detected events (OSA, OSA, or MSA).

5. Conclusions

In this study, a new algorithm is developed for automated detection of sleep apnea events using single channel data from oronasal thermal airflow sensor (FlowTH). The algorithm leverages AASM recommendations by defining apnea using relative changes in the oronasal airflow signal with respect to the evolving respiratory airflow baseline. A novel approach is developed to represent the respiratory airflow baseline based on two features: The inter-breath intervals and breath amplitudes. In order to detect apneic events, we considered a probabilistic view for classification using a GMM modeling framework. The proposed framework showed a significantly improved apnea detection performance for different apnea types and severity conditions as opposed to a rule-based classifier that uses the same input features as well as two previous methods for automated detection using the same respiratory source. The proposed modeling framework achieved an overall summary performance of $TPR = 88.5\%$, $TNR = 82.5\%$, and $AUC = 86.7\%$.

Author Contributions: Conceptualization, H.E. and J.K.; methodology, H.E and J.K.; software, J.K.; validation, H.E. and J.K; formal analysis, J.K.; investigation, H.E., J.K., and M.R.; resources, H.E., J.K., D.T., and S.K.R.; data curation, H.E. and J.K.; writing—original draft preparation, H.E.; writing–review and editing, H.E, J.K., and M.R; visualization, H.E and J.K.; supervision, D.T., S.K.R., and C.-H.C.; project administration, D.T. and S.K.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Somers, V.K.; White, D.P.; Amin, R.; Abraham, W.T.; Costa, F.; Culebras, A.; Daniels, S.; Floras, J.S.; Hunt, C.E.; Olson, L.J.; et al. Sleep apnea and cardiovascular disease: An American heart association/American college of cardiology foundation scientific statement from the American heart association council for high blood pressure research professional education committee, council on clinical cardiology, stroke council, and council on cardiovascular nursing in collaboration with the national heart, lung, and blood institute national center on sleep disorders research (national institutes of health). *J. Am. Coll. Cardiol.* **2008**, *52*, 686–717. [PubMed]
2. Botros, N.; Concato, J.; Mohsenin, V.; Selim, B.; Doctor, K.; Yaggi, H.K. Obstructive sleep apnea as a risk factor for type 2 diabetes. *Am. J. Med.* **2009**, *122*, 1122–1127. [CrossRef] [PubMed]
3. Mandal, S.; Kent, B.D. Obstructive sleep apnoea and coronary artery disease. *J. Thorac. Dis.* **2018**, *10*, S4212. [CrossRef] [PubMed]
4. Sharma, S.; Culebras, A. Sleep apnoea and stroke. *Stroke Vasc. Neurol.* **2016**, *1*, 185–191. [CrossRef] [PubMed]

5. Quan, S.; Gillin, J.C.; Littner, M.; Shepard, J. Sleep-related breathing disorders in adults: Recommendations for syndrome definition and measurement techniques in clinical research. editorials. *Sleep* **1999**, *22*, 662–689. [CrossRef]
6. American Academy of Sleep Medicine. *The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications*; American Academy of Sleep Medicine: Westchester, IL, USA, 2007.
7. Dempsey, J.A.; Veasey, S.C.; Morgan, B.J.; O'Donnell, C.P. Pathophysiology of sleep apnea. *Physiol. Rev.* **2010**, *90*, 47–112. [CrossRef] [PubMed]
8. Zhang, J.; Zhang, Q.; Wang, Y.; Qiu, C. A real-time auto-adjustable smart pillow system for sleep apnea detection and treatment. In Proceedings of the 2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Philadelphia, PA, USA, 9–11 April 2013; pp. 179–190.
9. De Chazal, P.; Penzel, T.; Heneghan, C. Automated detection of obstructive sleep apnoea at different time scales using the electrocardiogram. *Physiol. Meas.* **2004**, *25*, 967. [CrossRef]
10. Patil, S.P.; Schneider, H.; Schwartz, A.R.; Smith, P.L. Adult obstructive sleep apnea: Pathophysiology and diagnosis. *Chest J.* **2007**, *132*, 325–337. [CrossRef]
11. Berry, R.B.; Brooks, R.; Gamaldo, C.E.; Harding, M.; loy, R.M.; Marcos, C.; Vaughn, B.V. *The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications*; Version 2.6.0; American Academy of Sleep Medicine: Darien, IL, USA, 2020.
12. Berry, R.B.; Budhiraja, R.; Gottlieb, D.J.; Gozal, D.; Iber, C.; Kapur, V.K.; Marcus, C.L.; Mehra, R.; Parthasarathy, S.; Quan, S.F.; et al. Rules for scoring respiratory events in sleep: Update of the 2007 AASM Manual for the Scoring of Sleep and Associated Events. *J. Clin. Sleep Med.* **2012**, *8*, 597–619. [CrossRef]
13. Agarwal, R.; Gotman, J. Computer-assisted sleep staging. *IEEE Trans. Biomed. Eng.* **2001**, *48*, 1412–1423. [CrossRef]
14. Flemons, W.W.; Littner, M.R.; Rowley, J.A.; Gay, P.; Anderson, W.M.; Hudgel, D.W.; McEvoy, R.D.; Loube, D.I. Home diagnosis of sleep apnea: A systematic review of the literature: an evidence review cosponsored by the American Academy of Sleep Medicine, the American College of Chest Physicians, and the American Thoracic Society. *CHEST J.* **2003**, *124*, 1543–1579. [CrossRef]
15. de Almeida, F.R.; Ayas, N.T.; Otsuka, R.; Ueda, H.; Hamilton, P.; Ryan, F.C.; Lowe, A.A. Nasal pressure recordings to detect obstructive sleep apnea. *Sleep Breath.* **2006**, *10*, 62–69. [CrossRef]
16. Khandoker, A.H.; Gubbi, J.; Palaniswami, M. Automated scoring of obstructive sleep apnea and hypopnea events using short-term electrocardiogram recordings. *Inf. Technol. Biomed. IEEE Trans.* **2009**, *13*, 1057–1067. [CrossRef] [PubMed]
17. Whitney, C.W.; Gottlieb, D.J.; Redline, S.; Norman, R.G.; Dodge, R.R.; Shahar, E.; Surovec, S.; Nieto, F.J. Reliability of scoring respiratory disturbance indices and sleep staging. *Sleep* **1998**, *21*, 749–757. [CrossRef]
18. Bennett, J.; Kinnear, W. *Sleep on the Cheap: The Role of Overnight Oximetry in the Diagnosis of Sleep Apnoea Hypopnoea Syndrome*; BMJ Publishing Group Ltd.: London, UK, 1999.
19. Koley, B.; Dey, D. Automated detection of apnea and hypopnea events. In Proceedings of the 2012 Third International Conference on Emerging Applications of Information Technology, Kolkata, India, 30 November–1 December 2012; pp. 85–88.
20. Gutiérrez-Tobal, G.C.; Álvarez, D.; Marcos, J.V.; Del Campo, F.; Hornero, R. Pattern recognition in airflow recordings to assist in the sleep apnoea–hypopnoea syndrome diagnosis. *Med Biol. Eng. Comput.* **2013**, *51*, 1367–1380. [CrossRef]
21. Gutiérrez-Tobal, G.; Hornero, R.; Álvarez, D.; Marcos, J.; Del Campo, F. Linear and nonlinear analysis of airflow recordings to help in sleep apnoea–hypopnoea syndrome diagnosis. *Physiol. Meas.* **2012**, *33*, 1261. [CrossRef]
22. Gutiérrez-Tobal, G.C.; Álvarez, D.; del Campo, F.; Hornero, R. Utility of adaboost to detect sleep apnea-hypopnea syndrome from single-channel airflow. *IEEE Trans. Biomed. Eng.* **2016**, *63*, 636–646. [CrossRef] [PubMed]
23. Nigro, C.A.; Dibur, E.; Aimaretti, S.; González, S.; Rhodius, E. Comparison of the automatic analysis versus the manual scoring from ApneaLink™ device for the diagnosis of obstructive sleep apnoea syndrome. *Sleep Breath.* **2011**, *15*, 679–686. [CrossRef] [PubMed]
24. Nakano, H.; Tanigawa, T.; Furukawa, T.; Nishima, S. Automatic detection of sleep-disordered breathing from a single-channel airflow record. *Eur. Respir. J.* **2007**, *29*, 728–736. [CrossRef]

25. BaHammam, A.; Sharif, M.; Gacuan, D.E.; George, S. Evaluation of the accuracy of manual and automatic scoring of a single airflow channel in patients with a high probability of obstructive sleep apnea. *Med Sci. Monit. Int. Med J. Exp. Clin. Res.* **2011**, *17*, MT13. [CrossRef]
26. Lin, Y.Y.; Wu, H.T.; Hsu, C.A.; Huang, P.C.; Huang, Y.H.; Lo, Y.L. Sleep apnea detection based on thoracic and abdominal movement signals of wearable piezoelectric bands. *IEEE J. Biomed. Health Inf.* **2016**, *21*, 1533–1545. [CrossRef]
27. Vaughn, C.M.; Clemmons, P. Piezoelectric belts as a method for measuring chest and abdominal movement for obstructive sleep apnea diagnosis. *Neurodiagn. J.* **2012**, *52*, 275–280.
28. Avcı, C.; Akbaş, A. Sleep apnea classification based on respiration signals by using ensemble methods. *Bio-Med Mater. Eng.* **2015**, *26*, S1703–S1710. [CrossRef]
29. Azimi, H.; Gilakjani, S.S.; Bouchard, M.; Goubran, R.A.; Knoefel, F. Automatic apnea-hypopnea events detection using an alternative sensor. In Proceedings of the 2018 IEEE Sensors Applications Symposium (SAS), Seoul, Korea, 12–14 March 2018; pp. 1–5.
30. Penzel, T.; McNames, J.; De Chazal, P.; Raymond, B.; Murray, A.; Moody, G. Systematic comparison of different algorithms for apnoea detection based on electrocardiogram recordings. *Med Biol. Eng. Comput.* **2002**, *40*, 402–407. [CrossRef] [PubMed]
31. De Chazal, P.; Heneghan, C.; Sheridan, E.; Reilly, R.; Nolan, P.; O'Malley, M. Automated processing of the single-lead electrocardiogram for the detection of obstructive sleep apnoea. *IEEE Trans. Biomed. Eng.* **2003**, *50*, 686–696. [CrossRef] [PubMed]
32. Bsoul, M.; Minn, H.; Tamil, L. Apnea MedAssist: Real-time sleep apnea monitor using single-lead ECG. *IEEE Trans. Inf. Technol. Biomed.* **2010**, *15*, 416–427. [CrossRef] [PubMed]
33. Burgos, A.; Goñi, A.; Illarramendi, A.; Bermúdez, J. Real-time detection of apneas on a PDA. *IEEE Trans. Inf. Technol. Biomed.* **2009**, *14*, 995–1002. [CrossRef]
34. Magalang, U.J.; Dmochowski, J.; Veeramachaneni, S.; Draw, A.; Mador, M.J.; El-Solh, A.; Grant, B.J. Prediction of the apnea-hypopnea index from overnight pulse oximetry. *Chest J.* **2003**, *124*, 1694–1701. [CrossRef]
35. Alvarez, D.; Hornero, R.; Marcos, J.V.; del Campo, F. Multivariate analysis of blood oxygen saturation recordings in obstructive sleep apnea diagnosis. *IEEE Trans. Biomed. Eng.* **2010**, *57*, 2816–2824. [CrossRef]
36. Yadollahi, A.; Giannouli, E.; Moussavi, Z. Sleep apnea monitoring and diagnosis based on pulse oximetery and tracheal sound signals. *Med Biol. Eng. Comput.* **2010**, *48*, 1087–1097. [CrossRef]
37. Xie, B.; Minn, H. Real-time sleep apnea detection by classifier combination. *IEEE Trans. Inf. Technol. Biomed.* **2012**, *16*, 469–477. [CrossRef]
38. El Shayeb, M.; Topfer, L.A.; Stafinski, T.; Pawluk, L.; Menon, D. Diagnostic accuracy of level 3 portable sleep tests versus level 1 polysomnography for sleep-disordered breathing: A systematic review and meta-analysis. *Cmaj* **2014**, *186*, E25–E51. [CrossRef]
39. Fontenla-Romero, O.; Guijarro-Berdiñas, B.; Alonso-Betanzos, A.; Moret-Bonillo, V. A new method for sleep apnea classification using wavelets and feedforward neural networks. *Artif. Intell. Med.* **2005**, *34*, 65–76. [CrossRef]
40. Han, J.; Shin, H.B.; Jeong, D.U.; Park, K.S. Detection of apneic events from single channel nasal airflow using 2nd derivative method. *Comput. Methods Programs Biomed.* **2008**, *91*, 199–207. [CrossRef]
41. Selvaraj, N.; Narasimhan, R. Detection of sleep apnea on a per-second basis using respiratory signals. In Proceedings of the Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE, Osaka, Japan, 3–7 July 2013; pp. 2124–2127.
42. Ciołek, M.; Niedźwiecki, M.; Sieklicki, S.; Drozdowski, J.; Siebert, J. Automated detection of sleep apnea and hypopnea events based on robust airflow envelope tracking in the presence of breathing artifacts. *IEEE J. Biomed. Health Inf.* **2015**, *19*, 418–429. [CrossRef]
43. Koley, B.L.; Dey, D. Automatic detection of sleep apnea and hypopnea events from single channel measurement of respiration signal employing ensemble binary SVM classifiers. *Measurement* **2013**, *46*, 2082–2092. [CrossRef]
44. Koley, B.L.; Dey, D. Real-time adaptive apnea and hypopnea event detection methodology for portable sleep apnea monitoring devices. *IEEE Trans. Biomed. Eng.* **2013**, *60*, 3354–3363. [CrossRef] [PubMed]
45. Várady, P.; Micsik, T.; Benedek, S.; Benyó, Z. A novel method for the detection of apnea and hypopnea events in respiration signals. *Biomed. Eng. IEEE Trans.* **2002**, *49*, 936–942. [CrossRef]

46. Tian, J.; Liu, J. Apnea detection based on time delay neural network. In Proceedings of the 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Shanghai, China, 17–18 Januray 2005; pp. 2571–2574.
47. Norman, R.G.; Rapoport, D.M.; Ayappa, I. Detection of flow limitation in obstructive sleep apnea with an artificial neural network. *Physiol. Meas.* **2007**, *28*, 1089. [CrossRef]
48. Ozerov, A.; Lagrange, M.; Vincent, E. GMM-based classification from noisy features. In Proceedings of the International Workshop on Machine Listening in Multisource Environments (CHiME), Florence, Italy, 1 September 2011; pp. 30–35.
49. Kim, J.; ElMoaqet, H.; Tilbury, D.M.; Ramachandran, S.K.; Penzel, T. Time domain characterization for sleep apnea in oronasal airflow signal: A dynamic threshold classification approach. *Physiol. Meas.* **2019**, *40*, 054007. [CrossRef]
50. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
51. Thomas, E.; Temko, A.; Lightbody, G.; Marnane, W.; Boylan, G. Gaussian mixture models for classification of neonatal seizures using EEG. *Physiol. Meas.* **2010**, *31*, 1047. [CrossRef] [PubMed]
52. ElMoaqet, H.; Tilbury, D.M.; Ramachandran, S.K. Predicting oxygen saturation levels in blood using autoregressive models: A threshold metric for evaluating predictive models. In Proceedings of the 2013 American Control Conference, Washington, DC, USA, 17–19 June 2013; pp. 734–739.
53. ElMoaqet, H.; Tilbury, D.M.; Ramachandran, S.K. Evaluating predictions of critical oxygen desaturation events. *Physiol. Meas.* **2014**, *35*, 639. [CrossRef]
54. ElMoaqet, H.; Tilbury, D.M.; Ramachandran, S.K. Multi-step ahead predictions for critical levels in physiological time series. *IEEE Trans. Cybern.* **2016**, *46*, 1704–1714. [CrossRef]
55. Zweig, M.H.; Campbell, G. Receiver-Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine. *Clin. Chem.* **1993**, *39*, 561–577. [CrossRef]
56. Koley, B.; Dey, D. Adaptive classification system for real-time detection of apnea and hypopnea events. In Proceedings of the 2013 IEEE Point-of-Care Healthcare Technologies (PHT), Bangalore, India, 16–18 January 2013; pp. 42–45.
57. Canadian Agency for Drugs and Technologies in Health (CADTH). Portable monitoring devices for diagnosis of obstructive sleep apnea at home: Review of Accuracy, cost-effectiveness, guidelines, and coverage in Canada. *CADTH Technol. Overviews* **2010**, *1*.
58. Collop, N.A.; Anderson, W.M.; Boehlecke, B.; Claman, D.; Goldberg, R.; Gottlieb, D.J.; Hudgel, D.; Sateia, M.; Schwab, R. Clinical guidelines for the use of unattended portable monitors in the diagnosis of obstructive sleep apnea in adult patients. *J. Clin. Sleep. Med.* **2007**, *3*, 737–747.
59. Baron, K.G.; Duffecy, J.; Berendsen, M.A.; Mason, I.C.; Lattie, E.G.; Manalo, N.C. Feeling validated yet? A scoping review of the use of consumer-targeted wearable and mobile technology to measure and improve sleep. *Sleep Med. Rev.* **2018**, *40*, 151–159. [CrossRef]
60. Ruehland, W.R.; Rochford, P.D.; O'Donoghue, F.J.; Pierce, R.J.; Singh, P.; Thornton, A.T. The new AASM criteria for scoring hypopneas: Impact on the apnea hypopnea index. *Sleep* **2009**, *32*, 150. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Optimization of Warehouse Operations with Genetic Algorithms

Mirosław Kordos [1,*], Jan Boryczko [1], Marcin Blachnik [2] and Sławomir Golak [2]

[1] Department of Computer Science and Automatics, University of Bielsko-Biała, 43-340 Bielsko-Biała, Poland; jb054384@student.ath.edu.pl

[2] Department of Applied Informatics, Silesian University of Technology, 44-100 Gliwice, Poland; marcin.blachnik@polsl.pl (M.B.); slawomir.golak@polsl.pl (S.G.)

* Correspondence: mkordos@ath.bielsko.pl

Received: 7 June 2020; Accepted: 9 July 2020; Published: 13 July 2020

Abstract: We present a complete, fully automatic solution based on genetic algorithms for the optimization of discrete product placement and of order picking routes in a warehouse. The solution takes as input the warehouse structure and the list of orders and returns the optimized product placement, which minimizes the sum of the order picking times. The order picking routes are optimized mostly by genetic algorithms with multi-parent crossover operator, but for some cases also permutations and local search methods can be used. The product placement is optimized by another genetic algorithm, where the sum of the lengths of the optimized order picking routes is used as the cost of the given product placement. We present several ideas, which improve and accelerate the optimization, as the proper number of parents in crossover, the caching procedure, multiple restart and order grouping. In the presented experiments, in comparison with the random product placement and random product picking order, the optimization of order picking routes allowed the decrease of the total order picking times to 54%, optimization of product placement with the basic version of the method allowed to reduce that time to 26% and optimization of product placement with the methods with the improvements, as multiple restart and multi-parent crossover to 21%.

Keywords: warehouse optimization; genetic algorithms; crossover

1. Introduction

A large share of operating costs related to product storage is connected to order picking. Based on many studies, it has been established that about 60% of warehouse operation costs are the costs of picking up goods when completing orders [1]. As the speed of this operation is a decisive factor in the response time to customers' orders and is one of the factors contributing to their decision about choosing or not the same company at the next purchase, it seems that the role of the speed of order completion is even greater.

Thus, shortening the time of order picking is the most important and most beneficial factor in reducing the costs of operating the warehouse. It can be achieved without significant costs by optimizing the locations for particular products in a warehouse and then determining the fastest order completion routes in the optimized product placement.

Some aspects of the optimization seems obvious, for example that items that are often ordered together should be placed close to each other, and frequently purchased items should be located close to the delivery point. However, for the case of discrete variables, considered in this paper, where the location sizes are fixed (some goods, e.g., storage of the sand, gravel, and so forth, can be expressed by continuous variables, but this problem is not considered here), with N items in the warehouse, the number of all possible their placements is $N!$. For $N = 100$ it gives $N! = 9.33 \times 10^{157}$, and even if the computer could analyze 1 billion permutations per second, it would 9.33×10^{157} years to find the best product placement. So in practice, designing manually an optimal product placement is impossible, as the number of possible arrangements significantly exceeds the possibilities of analyzing all solutions by humans, or even by a computer program, which tries all possible configurations.

In this paper, we present a complete, fully automated system based on artificial intelligence, particularly on genetic algorithms, which can overcome the limitations of the search space by an intelligent search. The system usually analyzes only several thousands to several tens of thousands possible product placements to find the optimal solution or at least a solution so close to the optimal one, that in practice it will not make any difference. Moreover, the system returns also the quickest order picking routes. The advantage of applying such a solution is speeding up the operations and thus reduction of warehouse operating costs (where typically 60% are the costs of order picking [1]) and the possibility to serve more customers by the same employees in the same time and thus to further increase the sales and profits.

Although this issue has been analyzed for a long time, and especially in recent years its intensive development has been taking place [2–4], we were not able to find in the literature a complete, accurate, fully automatic solution, which would consider the real distribution of orders in the optimization of product placements. All the papers we were able to find presented only some partial approaches, with big simplifications, for example that the route length is determined only by one product with the longest distance from the entrance [2], or that the user is responsible for re-allocating the products by placing the more frequently used closer to the entrance [5].

Moreover, frequently the list of products within a single order is quite long, especially in the warehouses which sell goods mostly to retailers, what makes the optimization even more important. The purpose of this paper is to fill the gap by presenting our solution and by discussing its particular aspects and their influence on the accuracy and speed of the warehouse optimization process.

First we define the problem (Section 2), then we list the main points of our contribution (Section 3), next we come to the details in the following order: the literature review (Section 4), presentation of the proposed solution (Sections 5.1–5.4), experimental results (Section 6) and conclusions (Section 7).

2. Problem Statement

The product placement determines the locations (usually shelves) of particular products in the warehouse. We define the cost of a product placement as the sum of the shortest order picking routes over all orders included in the order list for this product placement (Equation (1)). The assumption behind that is that if robots are used to collect the orders, then they will exactly follow the shortest routes found. If humans pick the orders, they will in most cases follow the same routes, however, sometimes they may decide to change the path. This can be however treated as random process and thus cannot be taken into account in the optimization.

$$Cost = \sum_{n=1}^{N_{ord}} \cdot Route_{min}(n), \tag{1}$$

where N_{ord} is the number orders in the order list and $Route_{min}(n)$ is the shortest order picking (order completion) route found for the n-th order.

The problem to solve is to find a product placement with with as low cost as possible. In other words we need to minimize the cost given by Equation (1) by proper products placing at particular locations in a warehouse.

For that purpose we developed the product placement optimization algorithm, which we describe in the following sections.

The inputs to the algorithm are:

- warehouse layout in the form of transition costs between neighboring locations
- the list of orders

The outputs of the algorithm are:

- the optimized product placement in the warehouse
- shortest order picking routes for each order from the order list for the optimized product placement

Here we only present the main idea and the details about each input and output can be found in the subsequent sections. The order list contains all considered orders, which should be the orders expected in a certain future period of time. In most cases these can be the past orders, as we can expect that the future orders will have the same distribution of products. Otherwise these can be the past orders from a corresponding season of the last year or the predicted future orders. Each order consists of several (at least one) products. To complete the order, the locations of all the products that are included in the order must be visited and the products must be picked. Thus a sub-problem of the main problem of product placement optimization is to find the quickest (shortest) order completion route for each order. This is necessary, as the sum of the quickest order picking routes is the main objective that we want to minimize by optimizing the placement of particular products in the warehouse.

This sub-problem of finding the quickest order completion route is equivalent to the Traveling Salesman Problem (TSP). While, the whole optimization of product placement is a different, much more complex problem. The main differences are:

1. In product placement optimization, the task is to find optimal product locations. In TSP the task is to find the shortest route connecting all considered locations.
2. In product placement optimization, the cost function that we use is the sum of the shortest order picking routes for a given product placement (Equation (1)). In TSP the cost function is the length a single route connecting all considered locations.
3. Product placement optimization is a two-level nested task, where the main (outer) process is the product placement optimization itself and the inner process calculates the cost (fitness) of each considered product placement by finding the shortest order picking routes and then summing them to obtain the cost. TSP is a single level problem, without any nested optimization.
4. In product placement optimization the methods developed especially for TSP as nearest neighbor, or HGReX crossover in genetic algorithms cannot be used in the main process, because we do not look for any route, there does not exist any sequence of locations and there is no such a concept as "go to the next location", which the methods use.

3. Contribution

The main points of the contribution of this paper are:

1. The whole system design, with nested genetic-based optimization, where the cost of product placement is expressed by the sum of order completions times (Section 5.2).
2. An improved multi-parent version of the HGreX crossover for route optimization (Section 5.4.1).
3. Multiple fast restart procedure to increase the accuracy with minimal growth of the optimization time (Section 5.4.3).
4. Automatic selection of the order picking route calculation method to balance the optimization speed and accuracy (Section 5.3).
5. Input data format in the form of transition costs between neighboring locations, what joins maximal accuracy with minimal user effort for the data preparation Section 5.1.
6. Caching of fitness values (Section 5.4.2).

4. Related Works

4.1. Warehouse Planing and Operations

A lot of literature positions were devoted to warehouse operations. Van Gils et al. [4] provided a review and classification of the scientific literature investigating combinations of tactical and operational order picking planning problems. Grosse et al. [6] analyzed human factors in order picking. Dijkstra and Roodbergen [7] discussed predetermined order picking sequences, including various layouts of the warehouse and its aisles. They also discussed a dynamic programming approach that determines storage location assignments for those layouts, using the route length formulas and optimal properties. Bolaños Zuñiga [3] presented a formal mathematical model for simultaneously determining storage location assignment and picker routing, considering precedence constraints based on the weight of the products and location for each product in a general warehouse. Bartholdi [1] in his book presented practical considerations for warehouse planning and construction. Rakesh [8] discussed methods that determine optimal lane depth, number of storage levels, and other parameters of warehouse layout to minimize space and material handling costs. The book of Davarzani [9] discussed warehouse planning, technology, equipment, human resource management, connections to other department and companies. Zunic [10] considered various warehouse designs, especially the V-shape isles and calculated the order picking routes for these designs. Dharmapriya [11] discussed the use of simulated annealing for the warehouse layout optimization taking into account the total demand and traveling cost, but without considering the co-existence of various products in the same orders.

4.2. Genetic Algorithms in Warehouse Optimization

Artificial intelligence methods, in particular genetic algorithms, are solutions that have numerous successful implementations and that have been rapidly gaining popularity in recent years and were applied also to warehouse operation optimizations [12–14].

Genetic algorithms have two important advantages: fast intelligent search used to find the product placement and a global cost function.

Their first advantage is that due to intelligent searching, genetic algorithms do not need to analyze all solutions (all possible permutations of product locations), which is impossible due to their number. They usually analyze only a few thousands up to few hundred thousands of product placements (and not all $N!$ possibilities) to find the solution. Although genetic algorithms do not guarantee finding the optimal solution every time, the solutions found are close enough to the optimal solution so that in practice this does not make a significant difference.

Their second advantage is that using genetic algorithms, we do not have to define ourselves the rules that characterize good solutions. This is a very important, because usually we do not know how to define the rules optimally, and we only have some intuitive knowledge (e.g., goods often purchased together should be close to each other). However, the expression of this knowledge in the strict mathematical form is impossible because of the complexity of the system and the frequently opposite optimums of various order completions. With genetic algorithms it is enough to formulate the cost function, which is expressed here as the sum of all order picking route lengths or as the total time required to complete all orders from a certain period.

In genetic algorithms, the problem is coded using arrays called chromosomes by analogy with encoding in the chromosomes of biological organisms [12,13]. Each product placement encoded by a chromosome represents one solution (one individual). Initially, a pool of random solutions is generated (in each solution the products are randomly assigned to the locations in a warehouse). Then an intelligent search is applied with the help of three basic operations—selection, crossover and mutation. The selection mechanism selects individuals for the crossover operator. It is organized by analogy to the biological process, where the better individuals have a higher probability of becoming parents and exchanging information to create offspring.

The crossover operator generates a new individual (child) from the existing ones (parents). In this way it allows to combine the information encoded in the chromosomes of different individuals into one new individual. The mutation mechanism exchanges the values between some positions in an individual chromosome. Then selected individuals are promoted to the next generation. The process is repeated iteratively as long as the satisfactory solution is found or as long as the improvement in the solutions is still occurring.

To sum up, it should be stated that the use of genetic algorithms or other evolutionary optimization methods in applications to warehouse systems, including the optimization of the distribution of goods and order picking routes can bring measurable benefits to companies using these solutions, accelerating their work and reducing operating costs.

Although the idea of applying genetic algorithms to warehouse optimization or order picking route optimization was presented in some literature positions, we have not found a complete automatic solution, which considers the order distribution, as we present in this paper.

As the optimization of the order picking route (which is equivalent to the traveling salesman problem— see Section 2) is a much simpler problem than the optimization of product placement in the warehouse, as it can be expected, much more papers are dedicated to the order picking route optimization and only few papers discuss the product placement optimization. Below we shortly present some of them.

Wang [5] applied genetic algorithms to optimize a fitness function consisting of three weighed terms—the turnover of goods, the gravity center of storage racks and the relevance of the goods. Wei [15] used a similar approach with genetic algorithm with PMX crossover and the fitness were defined by the terms of the aisle access, the weight of the items and the dimensions of the shelves.

Avdeikins and Savrasovs [2] applied genetic algorithms to warehouse optimization using order crossover (OX). Each individual in the population represented a warehouse layout. Each gene was a unique item. In their solution the fitness of each individual was calculated as the sum of maximal picking distance for each order—$fitness(i) = \sum O_k d_{max}(I)$, where O was the order from the set 1, 2, ..., k and $d_{max}(I)$ was the distance to furthest picking position. Distances were expressed by an integer value that for the first item was 1, for second 2, and so forth, increasing by 1 from one SKU to another. Using such fitness function moves the most frequently sold items closer to the warehouse entry, but this does not place in neighboring locations items that are frequently sold together, as this solution did not consider this aspect, neither it determines the real order picking routes and thus this cannot be considered a complete solution.

As the solution presented by Avdeikins and Savrasovs [2] may at the first glance seem similar to our solution, it is worth pointing out the differences. In our solution the fitness is calculated as a sum of all order picking route lengths. This is a fundamental difference between their work and our solution, as this allows us to minimize the time of the real order picking operations and thus for obtaining very accurate solutions, which also minimizes the distances between items contained in one order. The next difference is, that we use real transition costs (or real distances) and not an approximation by increasing the distance always by one, as was used by them. The next difference is that our solution determines as well the product placement as the fastest order picking routes. Moreover, we use newer, effective crossover operators and propose a lot of improvements to accuracy and speed of the optimization.

4.3. Crossover Operators

Proper design of the crossover operator is a crucial factor in genetic algorithm performance. In this subsection we review the crossover operators and present in detail the AEX and HGreX operators, which are used in our solution.

Crossover allows to combine together the most valuable information from two or more different chromosomes (parents) into one chromosome (child) that can represent a better solution than its parents. For that kind of problems, where each item can occupy only one location at a time and each location must be occupied by one item, as order picking route optimization or product placement optimization in a warehouse, special crossover operators must be used to ensure that there will be no duplicate elements and that each element will be present in the newly created individual. Several such crossover operators have been developed.

Hassanat and Alkafaween [16] proposed several crossover operators, such as cut on worst gene crossover (COWGC) and collision crossover, and selection approaches, as select the best crossover (SBC). COWGC exchanges genes between parents by cutting the chromosome at the point that maximally decreases the cost. The collision crossover uses two selection strategies for the crossover operators. The first one selects this crossover operator from several examined operators, which maximally improves the fitness and the other one randomly selects any operator. The SBC algorithm applies multiple crossover operators at the same time on the same parents, and finally selects the best two offspring to enter the population. Hwang presented the order crossover (OX) and cycle crossover (CX) operators [17]. Tan proposed heuristic greedy crossover (HGreX) and its variants HRndX and RProX [18]. Other popular crossover operators comprise partially mapped crossover (PMX) edge recombination crossover (ERX) and alternating edges crossover (ERX) [19].

Several comparisons of the performance of these crossover operators can be found in the literature [19]. Based on these comparisons, the best performing methods for the traveling salesman problem were most frequently the variants of the HGreX crossover operator and the second best was the AEX crossover operator. For this reason we decided to start our approach from applying these two crossover operators for our warehouse optimization problem.

HGreX is only suitable for those kinds of problems, where the cost of transition between two positions can be defined. For example, it can be used to find the shortest order picking path, as we can define the distances (costs) between particular locations that contain the products from the orders and thus must be visited. However, it cannot be applied to optimization of the product placement in the warehouse, because the distribution of products is not directly related to any single order picking route, but to a whole set of different routes. Thus in this case, we can define the global goal, which is the minimization of sum of the lengths of all order picking routes, but we cannot express the cost between any two positions. AEX on the other hand does not use the transition cost and therefore can be applied also to the problems, where such cost cannot be defined, as the product placement optimization in a warehouse. First we will present the AEX operator and then the HGreX operator.

AEX creates the child from two parents by starting from the value, which is at the first position in the first parent. Then it adds this value, from the second parent, which in the second parent follows the value just taken from the first parent. Then again a value from the first parent that follows the value just selected from the second parent and so one. If this is impossible, because some element would repeat, then a random not selected so far element is chosen. In the presented example each position in the chromosome represents one location in the warehouse and each letter represents one product.

Let us assume we have two parents P1 and P2:

P1 = [A B C D E F G H]
P2 = [H A D B G F E C]

To create the child, We start from any position of the first parent P1. Let us start from A. Then we add this value which is in the second parent after A, so we add D

Ch = [A D _ _ _ _ _ _]
and the values remaining in the parents:
P1 = [~~A~~ B C ~~D~~ E F G H]
P2 = [H ~~A~~ ~~D~~ B G F E C]

Next we add to the child this value, which is in P1 after D, that is E

Ch = [A D E _ _ _ _ _]
and the values remaining in the parents:

P1 = [~~A~~ B C ~~D~~ ~~E~~ F G H]
P2 = [H ~~A~~ ~~D~~ B G F ~~E~~ C]

Next we add to the child this value, which is in P2 after E, that is C

Ch = [A D E C _ _ _ _]
and the values remaining in the parents:
P1 = [~~A~~ B ~~C~~ ~~D~~ ~~E~~ F G H]
P2 = [H ~~A~~ ~~D~~ B G F ~~E~~ ~~C~~]

Next we add to the child this value, which is in P1 after C, that is D. However, D has already been used, so this is not a valid choice. In such a case we select randomly one of the remaining values in P1. Let us select G.

Ch = [A D E C G _ _ _]
P1 = [~~A~~ B ~~C~~ ~~D~~ ~~E~~ F ~~G~~ H]
P2 = [H ~~A~~ ~~D~~ B ~~G~~ F ~~E~~ ~~C~~]

Next we add to the child this value, which is in P2 after G, that is F

Ch = [A D E C G F _ _]
and the values remaining in the parents:
P1 = [~~A~~ B ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ H]
P2 = [H ~~A~~ ~~D~~ B ~~G~~ ~~F~~ ~~E~~ ~~C~~]

Next we add to the child this value, which is in P1 after F, that is currently H

Ch = [A D E C G F H _]
and the values remaining in the parents:
P1 = [~~A~~ B ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ ~~H~~]
P2 = [~~H~~ ~~A~~ ~~D~~ B ~~G~~ ~~F~~ ~~E~~ ~~C~~]

And finally we add to the child this value, which is in P2 after H, that is currently B and the child becomes:

Ch = [A D E C G F H B]

The HGreX crossover operator works in similar way to AEX. The difference is, that it does not take alternatively the elements from both parents, but always chooses this element from the two parents to which the distance (cost) from the current element is shorter (lower).

Let us assume we have the same two parents P1 and P2:

P1 = [A B C D E F G H]
P2 = [H A D B G F E C]

To create the child, we start from any position of the first parent P1, let us start from A. Then we add this value which has lower transition cost (shorter distance) to A from the two values that appear directly after A in both parents, that is from B and D. Let us assume that the cost of going from A to B is 12, and from A to D is 15. So we choose B as the next position in the child.

Ch = [A B _ _ _ _ _ _]
and the values remaining in the parents:
P1 = [~~A~~ ~~B~~ C D E F G H]
P2 = [H ~~A~~ D ~~B~~ G F E C]

However, if the costs were—A to B: 18, and from A to D: 15, then we would have chosen D as the next position in the child. The conflicts are resolved identically as in AEX.

In Section 5.4.1 we introduce multi-parent versions of the crossover operators.

5. The Proposed Method

In this section, we describe the proposed genetic algorithm based method that optimizes the product placement and order picking routes in the warehouse. The purpose of the method, as described in Section 2, is to minimize the product placement cost given by Equation (1), this is to find such assignment of particular products to positions in the warehouse, which minimizes the sum of the shortest order picking routes over all orders from the order list. Thus, the optimization process consists of the outer procedure (main process), which is the product placement optimization (presented in Section 5.2) and the inner procedure (the sub-process), which is the order picking route optimization for each considered product placement (presented in Section 5.3).

5.1. Data Format and Problem Encoding

This subsection explains the input data format and the problem encoding in genetic algorithm chromosomes. At the end also calculation of the transition cost matrix is explained.

To determine the quality of a given product placement (see Algorithm 1), first we must find the order picking routes for each order (see Algorithm 2). To find them, we must calculate the transition cost matrix (costs of moving between each pair of locations in the warehouse), and to calculate the full matrix, the user must provide the transition costs (or distances) between the adjacent locations.

Figure 1 shows a sample layout of a very small warehouse, which we use to explain the data format and problem encoding. Of course the real warehouses, for which the methodology was created will be much larger. In this example the distances entered by user are shown with the color lines in Figure 1 and the distance matrix with these entries is shown in Table 1. As distances are symmetric (e.g., $dist(loc5, loc8) = dist(loc8, loc5)$), it is enough to fill the distances over the diagonal. The remaining distances (e.g., $dist(loc7, loc10)$) will be calculated automatically.

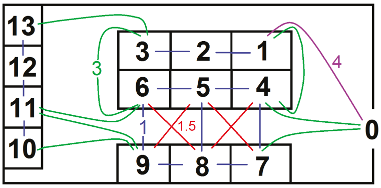


Figure 1. Sample warehouse structure used to explain the data format and problem encoding. Distances between neighboring locations: in blue the distances of 1 unit, in red of 1.5 unit, in green of 3 units, in violet of 4 units.

The warehouse layouts with product placements and the order picking routes are encoded in the chromosomes. Let us assume that the number of available products equals the number of locations in the warehouse, that is, 13 for this sample warehouse and the products names are: A,B,C,D,E,F,G,H,I,J,K,L,M. Let us also assume that there are three different orders, which occur with the same frequency and which consist of the following products:

Order1: A,B,C,D,E,F,G
Order2: G,H,I,J,K
Order3: A,B,K

For the product placement optimization we will encode the problem in a chromosome with 13 positions. At the beginning we generate a population of random individual chromosomes representing the product placements (see Algorithm 1). Let us assume, the 15th randomly generated individual looks as follows:

Layout15 = [G H E F I J A C D K B L M]

In this case the product G is at the location 1 in Figure 1, product H at location 2, and so on. We need to find the shortest order picking routes for each individual (each product placement) to calculate its fitness. For this purpose, we generate a population of random individual chromosomes representing the routes. There is always 0 (which represents the entrance) at the first position and the other positions are occupied by randomly ordered products from this order. Let us assume, the 12th randomly generated individual for Order3 looks as follows:

Order3-Route12 = [0 A K B 0]

The length of this route $lengthR$ is given by the formula:

$$lengthR = dist(loc0, loc7) + dist(loc7, loc10) + dist(loc10, loc11) + dist(loc11, loc0)$$

as in Layout15 $loc0$ is the entrance, product A at location 7, product K on location 10 and product B at location 11.

Table 1. The original distance matrix corresponding to the warehouse structure shown in Figure 1 containing only the values required by the algorithm.

.	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0		4			3			3						
1			1		3									
2				1										
3							3							3
4						1		1	1.5					
5							1	1.5	1	1.5				
6									1.5	1	3			
7									1					
8										1				
9											3	3		
10												1		
11													1	
12														1
13														

This input data format was specially designed in order to require minimal effort from the user entering the data, and at the same time to allow for maximal accuracy of calculations. Only the costs of transitions between neighboring locations are required in the input data. However, if the user wants to enter also the transition costs between some further locations, he is free to do it. The program preserves all distances entered by the user and only calculates the remaining distances.

The transition cost between locations can be entered as distance in meters, but also in seconds as the time needed to cover this distance. This takes into account that for example there is higher cost of covering the same distance vertically than horizontally or that turning around the corner requires more time than covering the same distance along a straight line and thus allowing to obtain higher accuracy of the order picking times. However, the units do not make any difference to the proposed method, which simply considers then as units of cost.

As the available plans of different warehouses can be in many different more or less usable formats, creating a separate software for preparation of the input data for each individual warehouse is no practical, as it would take more time than to enter the transition costs manually. The program does not need to know the geometrical structure of the warehouse. This is an additional advantage, because in this way much less work is required to prepare the input data.

Algorithm 1 Product Placement (PP) Optimization Process

Input 1: Warehouse layout in the form of transition costs between neighboring locations (see Table 1)

Input 2: The list of orders

Output 1: The optimized product placement (PP) in the warehouse

Output 2: Shortest order picking routes for each order for the optimized PP

```
 1: With Dijkstra algorithm calculate the full matrix D of transition costs between product locations
 2: for $k = 1$ to $numberOfProcessRestarts$ do
 3:   Generate the random population of $W$ PPs
 4:   if $numberOfProcessRestarts > 1$ then
 5:     $maxIterationPP = 10$ else $maxIterationPP = 1000$
 6:   end if
 7:   for $n = 1$ to $maxIterationPP$ do
 8:     (Calculate the fitness of $w$-th PP)
 9:     for $i = 1$ to $popSizePP$ do
10:       if $i$-th PP is already in cache then
11:         Retrieve $costPP(i)$ from cache
12:         Calculate the fitness $fitnessPP(i)$ of $i$-th PP according to Equation (3)
13:       else
14:         for $j = 1$ to $numOrdersWithDifferentItems$ do
15:           Run Algorithm 2 to find the shortest route for the $j$-th order and its length $lengthR(i, j)$
16:           Multiply $lengthR(i, j)$ by the number of orders, which contain the same items $N_{rep}(j)$
17:         end for
18:         Calculate cost of $i$-th PP as the sum of the lengths $lengthR$ of the all order best routes
19:         Calculate the fitness $fitnessPP(i)$ of $i$-th PP according to Equation (3)
20:         Update the cache
21:       end if
22:     end for
23:
24:     for $p = 1$ to $popSizePP$ do
25:       Select parents for each child PP
26:       Generate the child PP with the AEX crossover operator
27:     end for
28:     if $fitnessPP$ of the best PP has not improved for $NBI_{PP}$ iterations then
29:       break
30:     end if
31:     Sort the parent and child population of PPs
32:     promote $popSizePP$ PPs from best parents and best children to the next generation
33:     Apply mutation and update the cache for each mutated PP, which is not already in cache
34:   end for
35:   if $numberOfProcessRestarts > 1$ then
36:     Save the current population of PPs and their $fitnessPP$
37:   end if
38: end for
39: if $numberOfProcessRestarts > 1$ then
40:   set $numberOfProcessRestarts = 1$
41:   Restore the population of the best PP and continue the optimization from line 4
42: end if
43: Return the best PP and the corresponding set of the shortest order picking routes
```

Algorithm 2 Order Picking Route Optimization Process

```
Input 1: The matrix D of transition costs between product locations
Input 2: The list of items in the $j$-th orders
Input 3: The $i$-th product placement
Output 1: The shortest route for the $j$-th order completion $route_{min}(i,j)$
Output 2: The length of shortest route for the $j$-th order completion $lengthR_{min}(i,j)$

if number of products in $j$-th order <= $Threshold1$ then
    Determine $route_{min}(i,j)$ by evaluating permutations
else if number of products in $j$-th order <= $Threshold2$ then
    if $route_{min}(i,j)$ is in cache then
        Retrieve $route_{min}(i,j)$ and $lengthR_{min}(i,j)$ from cache
    else
        Generate the random population of $popSizeR$ routes
        for $k = 1$ to $maxIterationR$ do
            for $m = 1$ to $popSizeR$ do
                Calculate the length of each route $lengthR(i,m)$
            end for
            if the length of the shortest route did not decrease for $NBI_{route}$ iterations then
                Return $route_{min}(i,j)$ and $lengthR_{min}(i,j)$
            end if
            for $m = 1$ to $popSizeR$ do
                Calculate the fitness of the routes $fitnessR(i)$ for the selection operator according to
                Equation (5)
                Select parents for each child route
                Generate the child route with the crossover operator
            end for
            Apply mutation to routes
            Sort the parent and the child routes
            if the best route fitness has not improved for $NBI_{route}$ iterations then
                break
            end if
            promote $popSizeR$ routes from best parents and children to the next generation
        end for
        Update cache
    end if
else
    Determine the $route_{min}(i,j)$ with the Nearest Neighbor Algorithm
end if
Return $route_{min}(i,j)$ and $lengthR_{min}(i,j)$
```

To calculate the remaining transition costs between each pair of locations (line 1 in Algorithm 1). any algorithm that can do this can be used, for example Dijkstra [20], Floyd Warshall [21] or Bellman-Ford Algorithm [22]. We use Dijkstra Algorithm, because it is the fastest one, especially for sparse graphs (as is the case here), where each vertex is connected only with few other vertices. For a graph of v vertices (locations) and e connecting edges (transition costs), calculating all the distances with Dijkstra Algorithm with a priority queue has the complexity $O(v(e + vlogv))$, while the complexity of the two other algorithms is $O(v^3)$ and $O(ev^2)$.

Calculating the cost matrix with the Dijkstra Algorithm takes only a very small, practically negligible, fraction of the time of the whole optimization of the product locations. Moreover, once calculated, the cost matrix can be re-used for other product placement as long as the physical layout of the warehouse does not change. The A* Algorithm [23] can be faster than Dijkstra only when the approximate cost from the current to the target node can be assessed. However, in this problem, we are not able to assess the approximate cost, because we do not know the coordinates of particular locations, but only the transition cost between neighboring locations. Considering these two factors it is not justified to demand from the user preparing additional data with coordinates of each location in order to use the A* Algorithm, as in this case the gain of the CPU time (usually less than a second) would not compensate the lost of the user's time (usually several hours) spent on preparing the additional data.

5.2. Product Placement Optimization

In this subsection we present the algorithm used to optimize the placement of particular products in the warehouse in order to minimize the total time of completing the orders from the order list. The main optimization process is shown in pseudo-code in Algorithm 1 and as diagrams in Figures 2 and 3. The sub-process, which determines the shortest order picking routes is discussed in the next section.

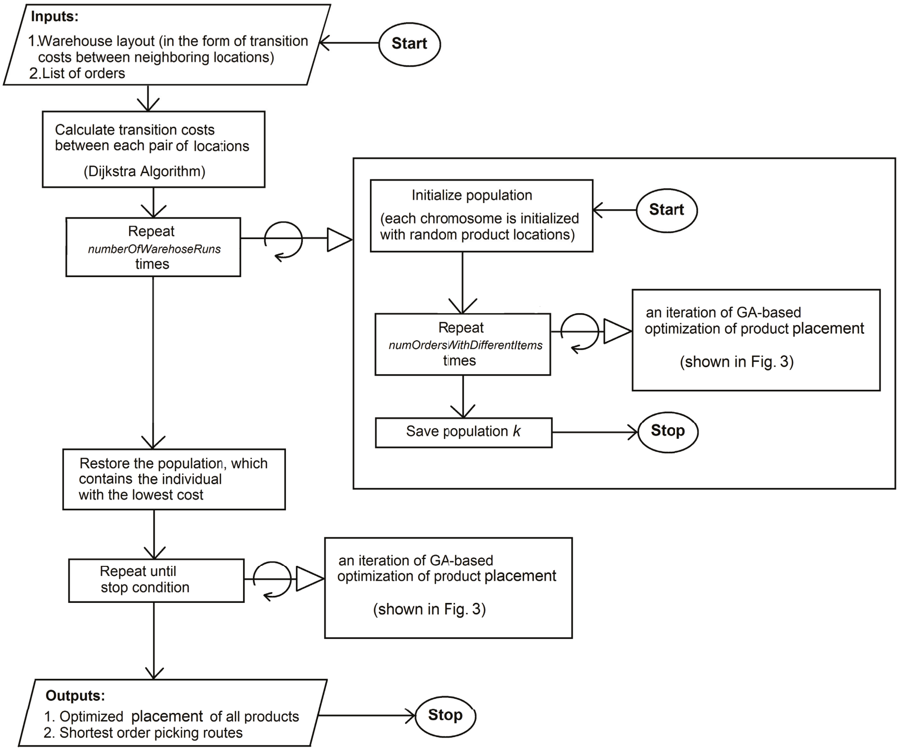

Inputs:
1.Warehouse layout (in the form of transition costs between neighboring locations)
2.List of orders
Start
Calculate transition costs between each pair of locations
(Dijkstra Algorithm)
Repeat
numberOfWarehoseRuns
times
Initialize population
(each chromosome is initialized with random product locations)
Start
Repeat
numOrdersWithDifferentItems
times
an iteration of GA-based optimization of product placement
(shown in Fig. 3)
Save population *k*
Stop
Restore the population, which contains the individual with the lowest cost
Repeat until stop condition
an iteration of GA-based optimization of product placement
(shown in Fig. 3)
Outputs:
1. Optimized placement of all products
2. Shortest order picking routes
Stop

Figure 2. Product Placement Optimization (main process).

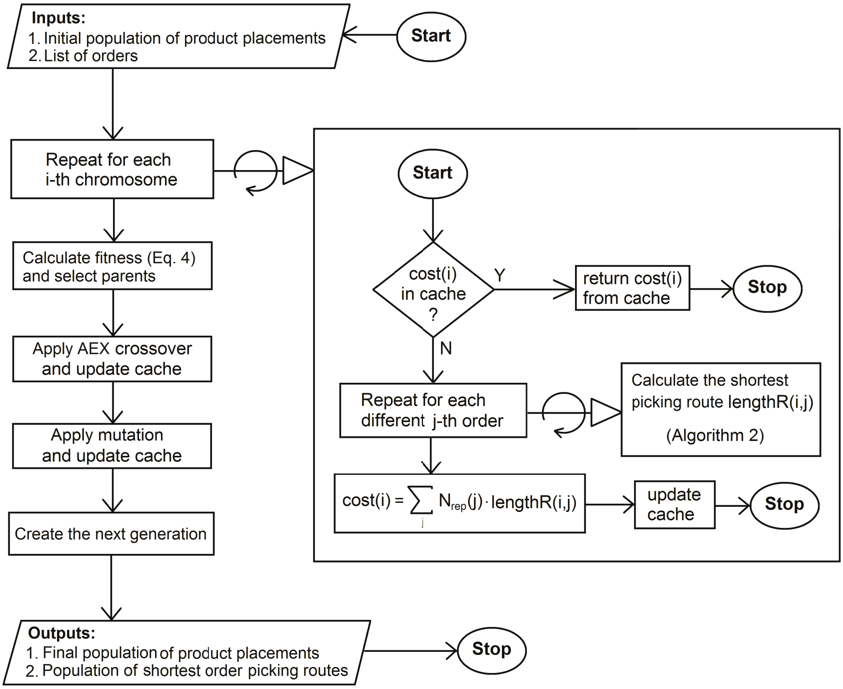

Inputs:
1. Initial population of product placements
2. List of orders
Start
Repeat for each i-th chromosome
Calculate fitness (Eq. 4) and select parents
Apply AEX crossover and update cache
Apply mutation and update cache
Create the next generation
Outputs:
1. Final population of product placements
2. Population of shortest order picking routes
Stop
Start
cost(i) in cache ?
Y
N
return cost(i) from cache
Stop
Repeat for each different j-th order
Calculate the shortest picking route lengthR(i,j) (Algorithm 2)
cost(i) = ∑_j N_rep(j)·lengthR(i,j)
update cache
Stop

Figure 3. Product Placement Optimization (inner block of the algorithm shown in Figure 2).

Now will explain the base version of the algorithm with $numberOfProcessRestarts = 1$ (line 2 in Algorithm 1) and in Section 5.4.3 we will explain the use and purpose of multiple process restarts ($numberOfProcessRestarts > 1$).

The process starts in line 1, where the Dijkstra algorithm calculates the full matrix **D** of transition costs (or distances) between product locations (see Section 5.1 for details). In line 3 the initial random population of product placements (PP) is generated. In line 7 the genetic algorithm starts the optimization. In line 10 the algorithm checks if any of the current individuals existed previously in current or any past epoch. If so the fitness of such an individual is not calculated but retrieved from the cache. In line 15 the shortest route for completing each different order for the current PP is calculated with Algorithm 2. Since we need to minimize the sum of order completion times, for each considered placement of products in the warehouse we need to calculate the time of each order completion and then add the times. To minimize the computational complexity of this step we group orders consisting of the same products together and assign to such an aggregated order a higher weight, which equals the number of single orders of which the aggregate order is composed. In line 18 the cost of the current PP is calculated, next the fitness is calculated from the cost and the cache is updated. In line 25 the parents for each child are selected and in the next line the child is created with the AEX crossover operator. If the fitness of the best PP has not improved for NBI_{PP} iterations (line 28) then the optimization is finished. In line 32 the promotion of children and best parents to the next generation takes place. In the next line the mutation is applied and the cache is updated. Finally in the last line the algorithm returns the best product placement and the corresponding set of the shortest order picking routes.

In order to keep the selection pressure constant and thus the exploration of the search space stronger at earlier stages and the convergence faster at later stages of the process [24] as well in product placement optimization as in the sub-process of order picking route optimization, we use use fitness function normalization. Thus the cost of a given product placement and the lengths of the order picking routes are not used directly as fitness values, but they are re-scaled. We used roulette wheel selection as well for the product placement optimization as for the order picking route optimization. According to Razali [25] and to our experiments, roulette wheel selection and tournament selection give comparable results. Also the selection pressure can be controlled in both (by number of candidates for a parent in tournament selection and by the shape of fitness function in roulette wheel selection). We chose roulette wheel selection for debugging purposes, as with this selection it was easier for to analyze the detailed algorithm behaviour and to fine-tune it.

The cost of the i-th product placement $costPP(i)$ expresses the sum of the shortest routes found for each order picking and is given by Equation (2):

$$CostPP(i) = \sum_{j=1}^{N_{diff}} N_{rep}(j) \cdot lengthR_{min}(j) \tag{2}$$

where N_{diff} is the number of different orders, $N_{rep}(j)$ is the numbers expressing how many times the j-th order is repeated on the order list and $lengthR_{min}(j)$ is the best (shortest) route found for the j-th order completion. The value $costPP$ is used to assess the progress and the results of the optimization (see Figure 3).

The fitness $fitnessPP(i)$ of the i-th product placement used by the roulette wheel selection is given by Equation (3):

$$fitnessPP(i) = c_1 + \frac{costPP_{max} - costPP(i)}{costPP_{max} - costPP_{min}}, \tag{3}$$

where c_1 is a coefficient (the lower c_1, the stronger preference for the individuals with lower cost), $costPP_{max}$ is the maximal cost, $costPP_{min}$ is the minimal cost and $costPP(i)$ is cost of the i-th product placement in the population.

The variable $fitnessPP$ take larger values for better individuals to ensure that better individuals have higher probability of being selected as parents, while $costPP$ take smaller values for better individuals, as they express the product placement cost, which equals the sum of lengths of the shortest routes.

We use dynamic mutation probability (the probability of exchanging some places in the chromosome), which increases gradually during the optimization. Also the probability of mutation is higher for the individuals with lower fitness. This minimizes the chance of disrupting a high-fitness individual and enhanced the exploratory role of low-fitness individuals. Lower mutation rates also allow for more effective caching of the fitness values (see Algorithm 1). The effectiveness of this approach was based on various observations [26]. We use two different mutation operators—Reverse Sequence Mutation (RSM) with and Partial Shuffle Mutation (PSM) with the probability of applying RSM being three times higher. The choice of these mutation operators is based on the experimental study by Otman et al. [27]. The total probability $mutationProb(i)$ of applying mutation to the i-th chromosome is expressed by Equation (4).

$$mutationProb(i) = (c_i\sqrt{iter} + c_n \cdot iter_{NBI})\frac{Fitness_{max}}{Fitness(i) + c_f}, \tag{4}$$

where c_i, c_n and c_f are coefficients, $iter$ is the current iteration (epoch) of the genetic algorithm, $iter_{NBI}$ is the number of iterations without improvement of the best individual. Default universal values of the coefficients for our purposes were experimentally set to $c_i = 0.00001$, $c_n = 0.00001$ and $c_f = 0.3$. Further refining of the mutation scheme, together with the mutation—crossover interactions is quite a complex issue and it will be one of our future research topics, when we will attempt to find optimal schemes for different situations.

5.3. Optimization of Order Picking Routes

As previously discussed, the task of optimization of order picking routes is a part of the optimization of product placement in the warehouse. The sum of the lengths of the order picking routes for a given product placement is its cost $costPP$—the lower the better (see Equation (1)). The process is presented in the pseudo-code in Algorithm 2 and in the diagram in Figure 4.

During the order picking route optimization the locations of products are constant. The product locations are changed by Algorithm 1 only before each round of order picking route optimization. As discussed in Section 5.1, the order picking route starts from the entrance, then visits all locations of products listed in the current order and returns to the warehouse entrance. The task is to optimize the sequence of visiting the locations to obtain the shortest route.

There are two main families of approaches to finding the shortest routes connecting a list of locations—the local search methods (e.g., nearest neighbor or k-opt [28]) and population methods (e.g., genetic algorithms or ant colony optimization [28]). In the Nearest Neighbor Algorithm, we always go from the current location to the nearest yet not visited location. In this way the algorithm implements local search. The local search guaranties finding the nearest location to the current location with 100% probability. However, the drawback of the local search approaches is that they do not include the global view of the situation. Although there was some research to improve these methods, the population methods still have the advantage of applying the global search. A sample route determined with the nearest neighbor, which shows the problems of this method, is shown in Figure 5.

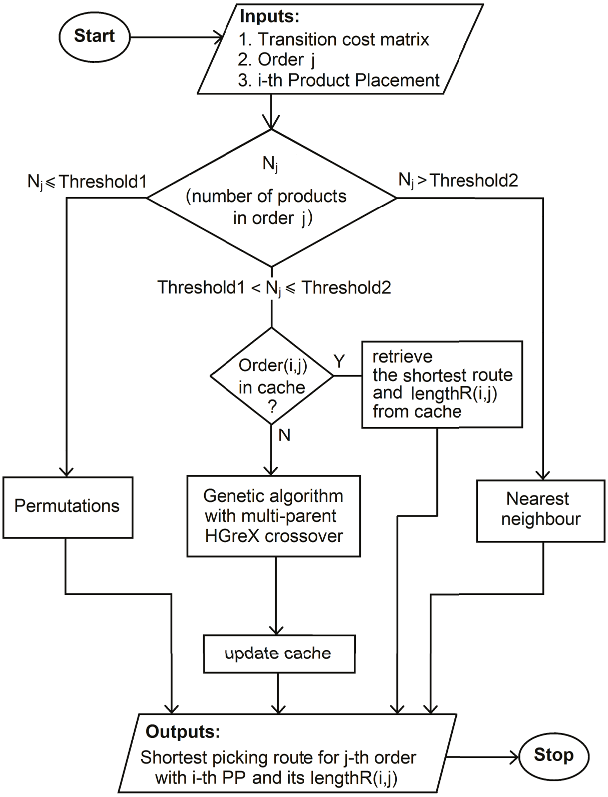

Start
Inputs:
1. Transition cost matrix
2. Order j
3. i-th Product Placement
N_j (number of products in order j)
N_j ≤ Threshold1
N_j > Threshold2
Threshold1 < N_j ≤ Threshold2
Order(i,j) in cache ?
Y
N
retrieve the shortest route and lengthR(i,j) from cache
Permutations
Genetic algorithm with multi-parent HGreX crossover
Nearest neighbour
update cache
Outputs:
Shortest picking route for j-th order with i-th PP and its lengthR(i,j)
Stop

Figure 4. Order picking route optimization.

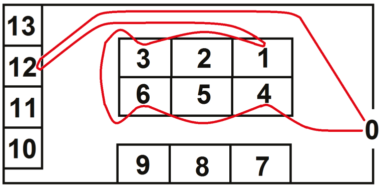


Figure 5. A sample route (in red) connecting the locations 0, 4, 6, 2, 1, 12 determined with the Nearest Neighbor Algorithm.

As will be shown in the experimental evaluations in Section 6 and as it is also known from previous studies [29], when the Nearest Neighbor Algorithm is used instead of genetic algorithms for that kind of problems, the calculation time can be dramatically reduced, however at the cost of worse results (on average 10% longer routes).

To determine the shortest route for completing each order, our method can use three different algorithms:

- Iterating through half of the possible permutations (as the transition cost matrix is symmetric—we can start the order completion from the end or from the beginning of the determined route, so we do not need to check all permutations, but only half of them). This method is 100% accurate, but is the slowest one expect for very short orders. As its complexity is $O(n!)$, it is impractical for orders above 10 positions and technically impossible to use for orders over 15 positions.
- Genetic algorithm with the multi-parent HGreX crossover operator. This method is faster than iterating over permutations, but does not guarantee finding the best solution, but only the solution close to the best one. (The basic version of the HGreX crossover is presented in Section 4.3 and the multi-parent extension in Section 5.4.1.) The fitness of the i-th order picking route is given by Equation (5). $fitnessR(i)$ takes larger values for better individuals to ensure that they have higher probability of being selected as parents, while $lengthR$ take smaller values for better individuals, as this value expresses the route length.

$$FitnessR(i) = c_2 + \frac{lengthR_{max} - lengthR(i)}{lengthR_{max} - lengthR_{min}}, \tag{5}$$

  where c_2 is a coefficient, which determined the strength of the selection (the lower c_2, the stronger preference for the individuals representing shorter routes) $lengthR_{max}$ is the maximal and $lengthR_{min}$ is the minimal length of the order picking route in the population. This ensures that the re-scaled proportion between the maximal and minimal fitness is constant during the optimization (see Section 5.2 for explanations).
- Nearest Neighbor Algorithm. This is the fastest method. It also does not guarantee finding the best solution, and in application to our problem it usually finds worse solutions than genetic algorithms.

To provide the optimal trade-off between the accuracy and the speed of the route optimization, the three above algorithms can be applied and the values *Threshold*1 and *Threshold*2 are used to determine, which particular algorithm will be used for a given order, depending on the number of products in the order (see Algorithm 2 and Figure 4).

The number of possible order picking routes is equal to the number of permutations of a k-element set, which is $k!$. If there are fewer than $k = 7$ products in a given order than it is faster to evaluate half of possible permutations (for $k = 6$ there are $6!/2 = 360$ various routes to examine.) than to use genetic algorithms. For $k = 7$ we need to evaluate $7!/2 = 2520$ permutations. On the other hand genetic algorithms are usually able to find the solution evaluating fewer routes (e.g., with population of 50 individuals and 10 iteration, what gives only 500 evaluations). However, genetic algorithms have additional time overhead for operations as selection, crossover, generating random numbers, and so forth. So for 7 products in the order the calculation time is comparable and for more than seven products, genetic algorithms are faster. Thus we propose to set $Threshold1 = 7$.

As can be seen in Section 6, the results obtained with genetic algorithms with multi-parent HGreX crossover are better those obtained with Nearest Neighbor Algorithms. On the other hand Nearest Neighbor Algorithm is faster than genetic algorithms. However, its speed advantage is much higher in a single optimization of the route, where it can be two orders of magnitude faster. In our system, when the route optimization is an iteratively performed sub-process of the product placement optimization, the differences in speed between these two methods is much lower, below one order of magnitude. There are two reasons for that. The first one is that there is implemented cashing of the already calculated routes (see details in Section 5.4.2). The caching overhead is comparable to the computational effort of Nearest Neighbor Algorithm, so it can only accelerate the genetic algorithm based route calculation. The second reason is that the time of running the main process (product placement optimization) is the same in both cases.

$Threshold2$ indicates above which number of products in the order Nearest Neighbor Algorithm should be used to optimize this order picking route. The recommendation to obtain the best results is to set $Threshold2$ to such high value that the Nearest Neighbor Algorithm will not be used at all (e.g., $Threshold2 = 1000$). However, if our data is very big and computational and time resources are limited, we can set $Threshold1 = 0$ and $Threshold2 = 0$ and thus only the Nearest Neighbor Algorithms will be used for the route optimization. Sometimes it happens that there are only very few long orders (e.g., two orders of 50 products and all remaining orders below 20 products), so these few orders will not have significant influence on the final product placement and we can use Nearest Neighbor Algorithm for them to accelerate the calculations and genetic algorithms for all other orders (in this case by setting for example, $Threshold2 = 30$).

5.4. Improvements and Accelerations of the Process

We use the following improvements to obtain better results and to accelerate the process: multi-parent crossover operator, order grouping, caching product placement costs and order picking routes of evaluated individuals, multiple restart, switching among permutations/genetic algorithms/nearest neighbor for route optimization, and parallelization of the process. In the following subsections we present particular improvements. Influence of these improvements on the obtained results is evaluated experimentally and presented in tables and figures in Section 6.

5.4.1. Multi-Parent Crossover Operators

As the use of multi-parent crossover operators can significantly accelerate (up to three times) the convergence speed of the classical genetic algorithms [30,31] (as well their single-objective as multi-objective version built upon the NSGA-II algorithm [32]), one of the ideas of this work was to apply the multi-parent approach to the crossover operators in the route and product placement optimization problems in hope that it can provide better results.

There is also another rationale behind increasing the number of parents in the HGreX crossover operator. In the Nearest Neighbor Algorithm the positions are added one by one to the route; each time the closest position is appended to the last position. In the extreme case, when we have a big population so that almost each possible two-element sequence exists in the population and the number of parents in the multi-parent HGreX crossover equals the population size - the so constructed genetic algorithm becomes equivalent to the nearest neighbor search. But on the other hand increasing the number of parents only a little bit, may add the local search component to the genetic algorithms and thus improve the results.

Let us assume that we will use four parents to create each child.

P1 = [A B C D E F G H]
P2 = [E G F H A C B D]
P3 = [G H A E B F C D]
P4 = [E F H D B A G C]

Let us start from the fist position in P1, this is from A. Let us assume that there are the following distance d(A,B) = 12, d(A,C) = 15, d(A,E) = 18, d(A,G) = 11. Since in this the distance d(A,G) is the smallest the next position in the child will be G.

Ch = [A G _ _ _ _ _ _]
and the values remaining in the parents:
P1 = [~~A~~ B C D E F ~~G~~ H]
P2 = [E ~~G~~ F H ~~A~~ C B D]
P3 = [~~G~~ H ~~A~~ E B F C D]
P4 = [E F H D B ~~A~~ ~~G~~ C]

Then Let us assume that there are the following distance d(G,H) = 12, d(G,F) = 8, d(G,H) = 7, Since in this the distance d(G,H) is the smallest the next position in the child will be H, and so on. Conflict resolving is implemented in the same way as in the two-parent version of the operator. In case of AEX we were appending the consecutive positions to the child sequentially from consecutive parents.

We conducted the experiments with various number of parents and the conclusion was that for this problem about 8 parents is the optimal number for the modified HGreX crossover operator. As a result of applying multiple parents for the HGreX crossover, about a two-fold reduction of the number of iterations was observed, but what is more important, also shorter order picking routes were obtained (see the experimental results in Section 6 for details). On the other hand, increasing the number of parents for the AEX crossover did not significantly change the results.

5.4.2. Caching Cost of Product Placements and Lengths of Order Picking Routes

In product placement optimization the computational effort of calculating cost $costPP$ (see Algorithm 1) of an given product placement and then based on it the fitness value of an individual is high, as it requires finding the shortest picking routes for all orders. Practically always either some parents are promoted to the next generation or some children are identical to some parents (even more in the final stages of the optimization). In this case, we do not calculate the cost of such an individual, but instead we directly assign the already calculated and cached cost of the previous identical individual (See Algorithm 1 and Figure 2). We also check the cache after mutation.

The situation with order picking route optimization with genetic algorithms is different. Here the computational effort of calculating the route length and determining the fitness of an individual is low and there is no use to implement caching for that. However, the cost of calculating the shortest route for an order (which may require several thousands calculations of route lengths represented by all individuals in all iteration of the optimization) is much higher and it makes a sense to implement cache here. Thus before calculating the shortest route for a given order $route_{min}(i,j)$ (See Algorithm 2 and Figure 4), the cache is checked if it already contains the route for the current order j, where all the positions of products in the warehouse were the same. To clarify this, if the whole product placement can be found in cache, the sub-process of route calculations is not invoked from the main process, as the cost of this product placement $costPP$ is retrieved from cache. However, if the product placement differs on some positions, the sub-process is invoked and it is checked for each order, if the positions in the product placement occupied by the products contained in the current orders already exist in the cache. If so, the route length $lengthR_{min}(i,j)$ is retrieved from the cache. Otherwise it is calculated and the cache is updated. The order cache is used only for route calculation with genetic algorithms. (See Algorithm 1 and Figure 2).

The caching is not used for the Nearest Neighbor Algorithm, as the time overhead for the cache is comparable to the time used by nearest neighbor. For the same reason the cashing is not used with very short orders, where the shortest route is determined by permutations.

The caching obviously does not influence the results of the optimization and only allows to accelerate it. It is also worth noticing that the caching is more effective at the later stages of the optimization, as at the beginning the individuals change rapidly. In our experiments the caching allowed to accelerate the product placement optimization several times (see Section 6).

5.4.3. Multiple Restart

It may take many iterations for genetic algorithms to converge to the optimal solution. However, the fastest progress occurs at the beginning of the optimization. Genetic algorithms use some random numbers and thus are a stochastic process and as a result different solutions can be found with consecutive runs of the optimization. We observed that in the product placement optimization the best approach is to run the optimization several times only for a few iterations and save the current population. Then the optimization will continue only with the population of the best solution. It is a reasonable approach, because most frequently the optimization, which starts as the best also ends as the best. Thus this method allows joining time efficient optimization with good results (see the experimental verification in Section 6).

5.4.4. Order Grouping

All orders containing the same set of products are grouped together into one order. In this way the optimal picking route for this order has to be determined only once. For the purpose of calculating the cost and fitness of a given product placement, the length of the obtained route is multiplied by the number of the orders consisting of the same products (see Algorithm 1).

5.4.5. Three Route Optimization Methods

As described in Section 5.3, for the optimal balance between calculation speed and accuracy of the results, the order picking route in Algorithm 2 can be optimized with permutations (only short routes), genetic algorithms or the Nearest Neighbor Algorithm.

5.4.6. Process Parallelization

Genetic algorithms scale well for parallel implementations in the cases, where the cost of calculating the fitness function is high, because in these cases there is no need for frequent communication among threads. It is exactly the case of product placement optimization, where using any number of CPU cores up to the number of individuals in the population results in practically a linear increase of performance in the function of the number of CPU cores. Moreover, if there are more CPU cores available than the population size, it makes sense to increase the population size, at least up to three times to use more CPU cores. Although after exceeding the optimal population size the scaling with the growth of CPU number is no longer linear, this implementation is very simple. The other alternative with few hundreds of available CPU cores is to parallelize the calculation of particular order picking routes, but since we did not have access to such computational resources, we were not able to verify the efficiency of this approach.

6. Experimental Results

In this section we experimentally evaluate the method and improvements presented in the previous sections.

We conducted the experiments with our own software, created in C# language. The source code and the data used in the experiments (warehouse plans with lists of corresponding orders) are available from the web page www.kordos.com/appliedsciences2020. Three of these warehouse structures (floor plans) and some sample orders for the warehouse $w3$ are additionally presented in Figures 6 and 7.

The following algorithms of order picking route optimization were evaluated—nearest neighbor, genetic algorithms with HGreX crossover, genetic algorithms with multi-parent HGreX crossover.

The following algorithms of product placement optimization are evaluated: genetic algorithms with AEX crossover, genetic algorithms with multi-parent AEX crossover, genetic algorithms with multi-parent AEX crossover and multiple restart.

As we could not find in literature a complete automatic solution for product placement optimization, which considers the order picking routes, as the solution presented here (see Sections 1 and 4.2), we obviously can not compare numerically our solution to other solutions on the same data.

First we evaluated the multi-parent modifications of the HGreX crossover operator to determine the optimal number of parents (see Sections 5.3 and 5.4.1). The results are presented in Table 2 and in Figure 8. Based on our tests the population sizes of about $N = 80$–120 allowed for the fastest convergence of the process (the lowest number of fitness value calculations). For larger populations, fitness function evaluations had to be performed more times to reach to the same results, so even if it took fewer iterations, the time to reach the results was longer [33]. However, if the populations were smaller it also required more evaluations of the fitness function and if the populations were too small, the convergence of the algorithm was impossible. Only for route optimization, when the number of products in the order was 20 or less, lower sizes of population were used and larger populations may be useful for longer chromosomes than those we used in the experimental evaluation.

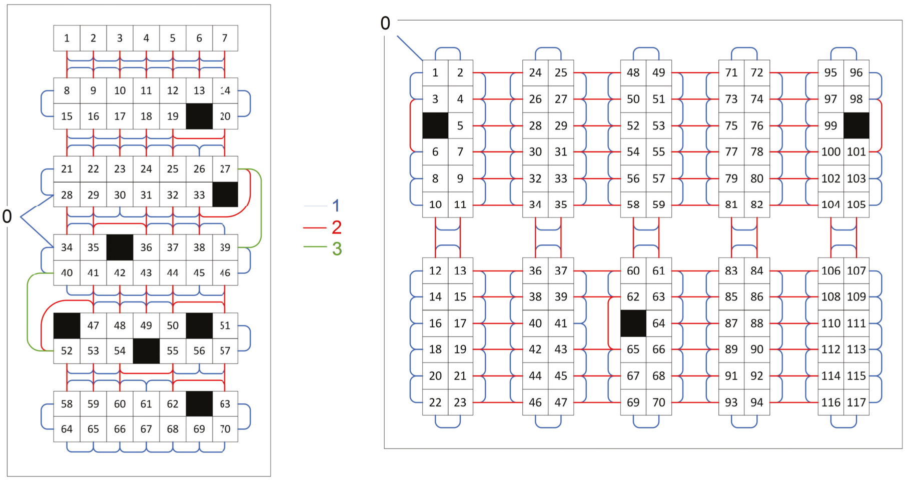


Figure 6. Samples warehouse structures (floor plans) of the warehouses w5 and w1 used in the experiments. Each numbered cell represents one product location. Blue lines show the distance of 1 unit, red lines of 2 units and green lines of 3 units.

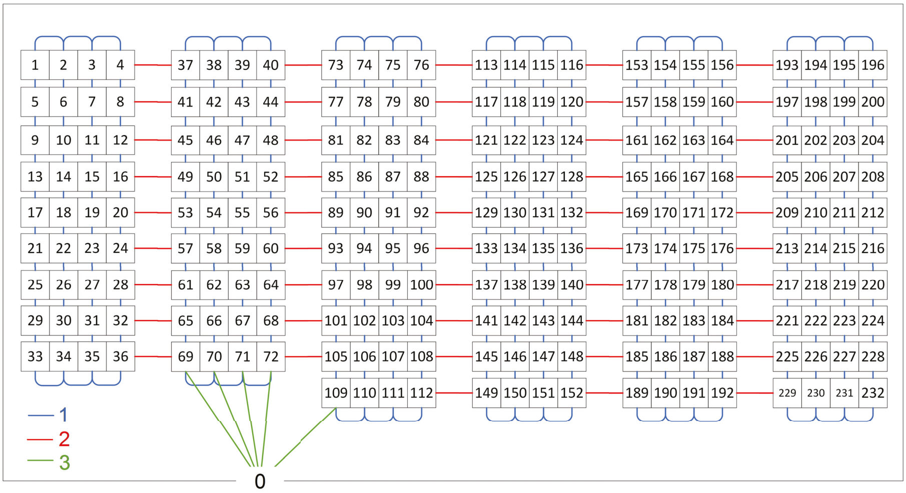


Figure 7. A sample warehouse structure (floor plan) of the warehouse w2 used in the experiments. Each numbered cell represents one product location. Blue lines show the distance of 1 unit, red lines of 2 units and green lines of 3 units.

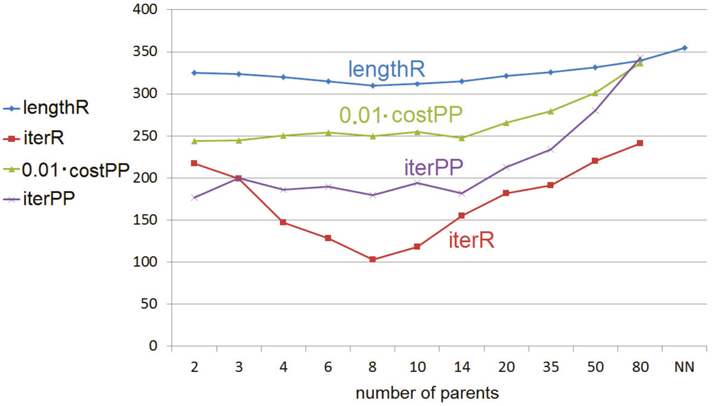


Figure 8. The obtained route length *lengthR* (the lower the better) and product placement cost *costPP* (the lower the better) and the number of iterations for route optimization *iterR* with MP-HGreX and for product placement optimization *iterP* with MP-AEX (graphical representation of the data from Tables 2 and 3).

The stopping criterion for experiments shown in Table 2 was 20 iterations without improvement of the best individual. The number of reported iterations is the number after which the best individual was found. As it can be seen, increasing the number of parents in the crossover operator definitely reduces the number of required iterations (up to two times for 8 parents in this case) and what is more important, allows for obtaining better fitness values. However, when using more parents than the optimal number, the optimization again slows down and using more than 20 parents also the obtained route lengths are beginning to deteriorate (to increase). As discussed in Section 5.3, for large number of parents the HGreX operator behaves almost like the nearest neighbor search method, and also its performance tends to the same value.

Table 2. A sample route length *lengthR* and number of iterations *iterR* to obtain this length for a modified multi-parent HGreX crossover operator for an order of 60 products with fixed product placement (averages of 10 runs). *NN* in the last column denotes the result obtained for the Nearest Neighbor Algorithm.

Number of Parents	2	3	4	6	8	10	14	20	35	50	80	NN
avg. *lengthR*	325	323	320	315	310	312	315	321	326	331	339	355
agv. *iterR*	217	199	147	128	103	124	155	182	191	220	243	-
std. dev. *lengthR*	3.5	3.8	3.9	3.4	3.4	3.8	4.0	5.4	5.2	9.2	8.8	-
std. *iterR*	46	52	39	26	33	31	38	68	70	79	67	-

The number of iterations used by the genetic algorithm with the HGreX crossover is comparable to that required by other modern crossover operators to find the shortest route for comparable population size [34]. Even if running the optimization for more iterations may find a little shorter route, there is usually no further gain for the product placement cost, as this only very rarely triggers the change of product locations. For the very rarely occurring orders it is enough to run the optimization for fewer epochs or to use Nearest Neighbor Algorithm, independently of the order length, because the quickest improvement occurs at the beginning of the optimization and the influence on the optimal product placement of very rare orders is also very low, as they are dominated by the more frequent orders.

Next we tested the usefulness of the multi-parent AEX crossover in product placement optimization. Since AEX does not use the cost of transitions between two elements, the parents for each element were chosen randomly. The results for a warehouse with 232 locations (chromosome length was 232 elements) are presented in Table 3. Based on the experiments we concluded that it is enough to use two-parent AEX crossover in product placement optimization, as increasing the number of parents did not cause any gain and if the number was 20 or more, the drop in the method effectiveness was observed.

Table 3. The product placement cost *costPP* as sum of order picking route lengths (the lower the better—see Equation 2) and number of iterations *iterP* to obtain this cost for a modified multi-parent AEX crossover operator for the warehouse size of 232 locations and a list of 80 orders, using an 8-parent HGreX for route optimization (averages of 10 runs).

Num. Parents	2	3	4	6	8	10	14	20	35	50
avg. *costPP*	24,397	24,456	25,054	25,410	24,969	25,511	24,785	26,607	27,949	30,014
best *costPP*	21,731	21,410	21,615	22,535	22,861	21,945	21,476	21,474	25,455	26,878
agv. *iterP*	177	200	186	190	180	202	182	213	234	280
std. dev. *costPP*	2586	2777	2158	1831	3165	3110	2660	3106	3250	2996
std. *iterP*	44	79	76	86	101	104	79	98	112	106

Multiple restart of the product placement optimization with AEX crossover (MR-AEX) proved quite useful (see Algorithm 1). In the last row of Table 4 the optimization was restarted 5 times and each time it was run for 10 iterations and then we continued only with the population of the best individual, as described in Section 5.4.3. It allowed not only to obtain lower cost, but also the standard deviation of the results was about twice lower.

In the experiments presented in Table 4 we used the population size of 100 individuals for product placement optimization. For order picking route optimization we used a size of 100 individuals if the number of items was 25 or more and four times the number of items for shorter orders. The reason for choosing that population size is based on this fact, that the main cost of genetic algorithms is the evaluation of the fitness function (especially for product placement optimization). The number of the fitness function evaluations can be expressed by the multiplying population size by the number of epochs. Using larger populations, we can obtain the same results in fewer epochs. For smaller populations we also need to increase the mutation rate. However, the dependence between the population size and number of required epochs is not linear and there exists an optimal population size, which allows for the lowest number of fitness function evaluations [33]. The number also depends on the problem and on other parameters of genetic algorithms. In our experiments, the minimum was usually obtained for the population sizes between 80 and 120 individuals for the number of locations in the warehouse between 60 and 300 and then it very slowly grew, but much slower than linearly, with the increase of the warehouse. The dependence was very flat around the minimum (changing the population size e.g., from 80 to 100 individuals did not make a statistically significant difference in the number of required fitness function evaluations). However, outside of this range the dependence was more significant and for example using 1000 individuals allowed to decrease the number of epochs only about 3 times, what effectively increased the number of fitness function evaluations about 3-fold. Moreover, when the population was too small, not only the process time increased, but the process also began to be unstable and frequently was not able to converge. For this reason, a population size of 100 individuals was a safer choice than of, for example, 80 individuals.

We used the default mutation coefficients in Equation (4): $c_i = 0.00001$, $c_n = 0.00001$, $c_f = 0.3$ as well in product placement as in route optimization.

Below we present some sample orders for the warehouse $w3$. The products in the orders are encoded by numbers, which are the products Ids (we cannot use letters as in the examples in previous sections, because there are not enough letters in the alphabet). The last number (N_{rep}) of each order shows how many times such order occurs in the order list, so its completion route length can be evaluated only once and then multiplied by N_{rep} while calculating the final product placement cost.

order1: 41 99 97 7 20 89 12 24 51 66 79 61 1 56 109 $N_{rep} = 40$
order2: 71 90 9 29 84 94 19 26 64 114 100 42 81 30 108 107 101 47 6 32 96 33 28 7 $N_{rep} = 20$
order3: 78 31 91 35 93 87 22 50 100 1 28 38 84 16 48 112 76 110 95 47 72 113 23 61 101 68 67 53 45 41 97 18 109 89 65 74 $N_{rep} = 3$
order4: 53 61 18 36 94 24 103 38 35 12 42 89 6 30 50 14 84 114 29 15 79 95 48 52 28 25 110 22 64 109 44 11 73 33 98 97 23 75 99 87 7 51 92 93 72 17 3 $N_{rep} = 1$

Table 4. The obtained product placement cost (the lower the better) as the sum of all order picking routes (see Equation (2)) for product placement and order picking route optimization methods with various improvements (see Section 5.4) for the six sample warehouses: w1, w2, w3, w4, w5, w6 with corresponding lists of orders, averaged over 10 optimization runs. The running times are presented in Table 5.

Prod. Plc. Optimiz.	Route Optimiz.	Cost	w1	w2	w3	w4	w5	w6	Cost vs. Rnd.	*t*-Test Wilcox.
random	random	average	29,346	49,374	42,942	14,156	16,934	40109	1.000	
		std.dev.	5445	9447	9050	2872	3747	7792	0.207	0.0001
random	N.Neighbor	average	16,931	32,879	27,501	9325	9102	20314	0.596	0.00001
		std.dev.	2399	4444	3686	1354	1220	2915	0.083	0.0104
random	HGreX	average	14,889	31,382	26,543	8950	8080	19060	0.575	0.00001
		std.dev.	2238	4139	3246	1047	995	2328	0.071	0.0102
random	MP-HGreX	average	14,342	30,660	26,004	8206	7917	18667	0.544	0.00001
		std.dev.	1529	3074	2042	684	712	1529	0.048	0.0001
AEX	N.Neighbor	average	6398	11,198	10,039	7274	3278	7400	0.262	0.00001
		std.dev.	418	747	619	525	212	498	0.018	0.0001
AEX	HGreX	average	5989	10760	9516	6048	3160	6898	0.238	0.00001
		std.dev.	550	809	826	453	253	544	0.019	0.0038
AEX	MP-HGreX	average	5694	10,767	9003	6000	3164	6778	0.227	0.00026
		std.dev.	382	649	533	403	221	441	0.015	0.0011
MR-AEX	MP-HGreX	average	5394	9676	8543	5312	3023	6428	0.213	0.00001
		std.dev.	171	339	289	174	101	226	0.007	

Table 4 presents the detailed results for six sample warehouse structures and order lists (this is the maximum number of warehouses, which can fit in one row of the table). MP-HGreX stands for Multi-Parent HGreX with 8 parents, MR-AEX is Multiple-Restart AEX with 5 restarts, saving the population afters 10 iterations, and then continuing with the population of the best individual (see Section 5.4.3 and Figure 2 for explanations). Table 5 presents the real running times of the optimization processes (including I/O operations and Dijkstra Algorithm).

Table 5. The real running time of the optimization processes in seconds using a computer with two Xeon X5-2696-v2 CPUs, averaged over 10 optimization runs for the experimental data presented in Table 4 and additionally for AEX/HGreX without cache (see Section 5.4.2).

Prod. Plc.	Route	w1	w2	w3	w4	w5	w6
random	random	0	0	0	0	0	0
random	N.Neighbor	1.4	2.0	2.0	1.0	1.0	1.8
random	HGreX	3.1	4.4	3.7	2.0	2.1	3.4
random	MP-HGreX	3.2	4.8	3.7	2.0	2.1	3.4
AEX	N.Neighbor	130	204	169	59	77	186
AEX	HGreX	744	1329	1121	423	495	1077
AEX	MP-HGreX	762	1334	1077	385	450	1133
MR-AEX	MP-HGreX	1441	1989	1667	762	827	1998
AEX/HGreX, no cache		3127	5678	4465	1710	2077	5225

The possible reduction of product placement cost depends on the character of the orders. The biggest improvement due to route optimization can be obtained for the orders containing long list of products. The highest improvement due to product placement optimization can be achieved when the orders frequently contain products of particular groups and the frequency with which particular products appear in the orders differs a lot. Thus the improvement possible to achieve is determined mostly by the properties of the orders. Thus, particular methods must be compared among each other for the same warehouse structure and for the same list of orders (This is similar, like in classification, where the possible accuracy depends on the dataset properties and various classifiers must be compared on the same data).

The *cost vs. rnd.* column in Table 4 contains the average relative reduction of the product placement cost calculated as $cost\ vs.\ rnd. = Average(Sum(F1p(w)/random(w)))$, where $w = 1 \ldots 6$ is the warehouse number. The last column contains statistical significance tests calculated on the whole data between two adjacent methods and therefore it is printed in-between rows of the compared methods. Since some persons prefer the T-test and others the Wilcoxon Signed Rank Test for this kind of data, we used both tests to satisfy everyone. As all the p-values in the last column of Table 4 are smaller than 0.05, it can be assumed that all the methods are significantly different from one another.

As can be seen in Figures 9 and 10, the best results are obtained for the multiple restart of genetic algorithms with AEX crossover operator (MR-AEX) for product placement optimization (see Section 5.4.3 and Algorithm 1) together genetic algorithm with multi-parent HGreX crossover operator (MP-HGreX) for order picking route optimization (see Figure 4).

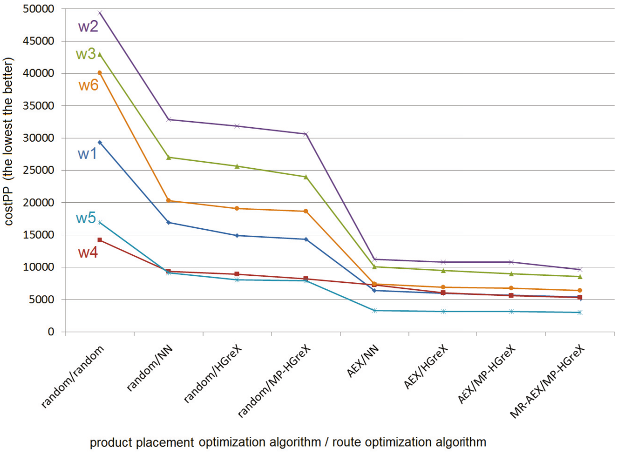


Figure 9. Graphical representation of the data from Table 4). On horizontal axis: the optimization methods with various improvements (see Section 5.4). On vertical axis: the product placement cost *costPP* (the lower the better) obtained with particular methods for the warehouses w1, w2, w3, w4, w5, w6 with corresponding lists of orders.

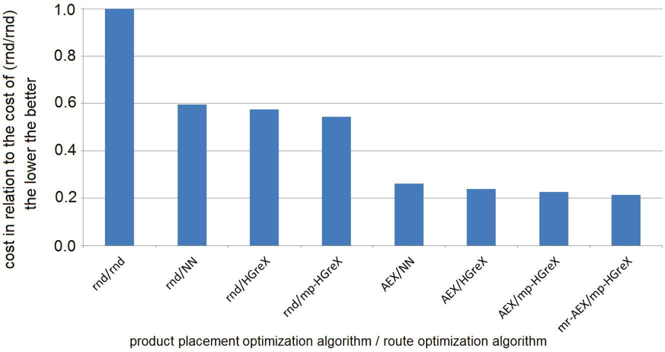


Figure 10. Comparison of the performance of the presented methods with various improvements (see Section 5.4). On horizontal axis: the optimization method. On vertical axis: the average obtained product placement cost (the lower the better) as percentage of the cost with the random product placement and random routes over the six warehouses with order lists presented in Table 4

7. Conclusions

Shortening the time of order picking is the most important and most beneficial factor in reducing the costs of operating the warehouse (where typically 60% are the costs are generated by order picking [1]). It can be achieved without significant investment by optimizing the locations for particular products in a warehouse and then determining the fastest order completion routes. As the search space of the solutions is enormous (9.3×10^{157} possible placements of 100 products, 3.1×10^{614} of 300 products) the problem cannot be analyzed by brute force methods. Thus we presented a complete, fully automatic system based on genetic algorithms, which due to applying intelligent search allows to find the optimal product placements within minutes or tens of minutes for that size of problem (depending on the computer hardware and process parameters). Even though it is not guaranteed that the optimal solution will be found with genetic algorithms, it is possible to find a very close solution to the optimal one, so that in practice it will not make a significant difference.

The presented system takes as inputs the warehouse structure (in the form of partial transition costs) and the list of orders and returns the optimal product placement and corresponding shortest order picking routes. Implementation of such a system can accelerate order picking and thus reduce the warehouse operating costs. This allows to serve more customers by the same number of employees in the same time and thus to further increase the sales and profits.

The experiments showed that using the multi-parent HGreX crossover improves the results, while for the AEX crossover adding more parents does not change its efficiency. The best results were obtained for the multiple restart genetic algorithm with AEX crossover operator (MR-AEX) for the product placement optimization process together genetic algorithm with multi-parent HGreX crossover operator (MP-HGreX) for order picking route optimization. The cost or route length caching can be used to accelerate the process. Additionally the Nearest Neighbor Algorithm can be used for route optimization to even more accelerate the process, but this is usually at the expense of a little worse result.

In the future works we are planing to implement other modifications to further improve the speed of the optimization and the quality of the obtained solutions. First we want to evaluate new crossover operators and mixtures of various operators. In the experimental comparison of Puljic [19] the mix of different crossover operators performed slightly better than HGreX. Also more advanced mixes of genetic operators were proposed [35,36]. However, we did not decide to use this approach because the implementation was definitely more complex. Instead we modified the HGreX to use multiple parents, what significantly improved the results. Łapa et al. [37] proposed the use of different operators (not only crossovers but also different mutations and other operators) for different individuals in standard genetic algorithms. We are going to adjust these approaches to the route and product placement optimizations and investigate various options.

The other branch of our future research refers to constraints in the genetic algorithm operations as in some warehouses such constraints may exist and may limit the possible locations of particular products. In the literature the typical approach to constraints in genetic algorithms is the use of penalty functions [38]. Sometimes also dominance-based methods are used [39]. However, we are going to implement it differently by embedding the mechanism directly into the crossover and mutation operators specific to that problem in order to be able to enforce the constraint effectively and to limit the computational complexity of the optimization.

Author Contributions: Conceptualization, M.K. and S.G.; Formal analysis, M.B. and S.G.; Funding acquisition, M.B. and S.G.; Investigation, M.K. and J.B.; Methodology, M.K., S.G. and J.B.; Software, M.K. and J.B.; Validation, M.B. and S.G.; Visualization, M.B. and S.G.; Writing—original draft, M.K., J.B., M.B. and S.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Silesian University of Technology project: BK-204/2020/RM4.

Acknowledgments: The authors want to thank Michał Krzyżowski, Łukasz Mysłajek, Antoni Kopeć and Jakub Gawęda for their help in collecting and preparing the data used in this study.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Bartholdi, J.J.; Hackman, S.T. Warehouse and Distribution Science. 2019. Available online: https://www.warehouse-science.com/book/index.html (accessed on 30 May 2020).
2. Avdeikins, A.; Savrasovs, M. Making Warehouse Logistics Smart by Effective Placement Strategy Based on Genetic Algorithms. *Transp. Telecommun.* **2019**, *20*, 318–324. [CrossRef]
3. Bolaños Zuñiga, J.; Saucedo Martínez, J.A.; Salais Fierro, T.E.; Marmolejo Saucedo, J.A. Optimization of the Storage Location Assignment and the Picker-Routing Problem by Using Mathematical Programming. *Appl. Sci.* **2020**, *10*, 534. [CrossRef]
4. Van Gils, T.; Ramaekers, K.; Caris, A.; De Koster, R. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *Eur. J. Oper. Res.* **2018**, *267*, 1–15. [CrossRef]

5. Wang, W.; Gao, J.; Gao, T.; Zhao, H. Optimization of Automated Warehouse Location Based on Genetic Algorithm. In Proceedings of the 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017), Sanya, China, 25–26 June 2017; pp. 309–313.
6. Grosse, E.H.; Glock, C.H.; Neumann, P.W. Human factors in order picking: A content analysis of the literature. *Int. J. Prod. Res.* **2016**, *55*, 1260–1276. [CrossRef]
7. Dijkstra, A.; Roodbergen, K. Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses. *Transp. Res. Part E* **2017**, *102*, 38–59. [CrossRef]
8. Rakesh, V.; Kadil, G. Layout Optimization of a Three Dimensional Order Picking Warehouse. *IFAC-PapersOnLine* **2017**, *48*, 1155–1160. [CrossRef]
9. Davarzani, H.; Norrman, A. Toward a relevant agenda for warehousing research: literature review and practitioners'. *Logist. Res.* **2015**, *8*, 1–18. [CrossRef]
10. Zunic, E.; Besirevic, A.; Skrobo, R.; Hasic, H.; Hodzic, K.; Djedovic, A. Design of Optimization System for Warehouse Order Picking in Real Environment. In Proceedings of the XXVI International Conference on Information, Communication and Automation Technologies, Sarajevo, Bosnia and Herzegovina, 26–28 October 2017; Volume 26.
11. Dharmapriya, U.; Kulatunga, A. New Strategy for Warehouse Optimization—Lean warehousing. In Proceedings of the 2011 International Conference on Industrial Engineering and Operations, Kuala Lumpur, Malaysia, 22–24 January 2011.
12. Affenzeller, M.; Wagner, S.; Winkler, S.; Beham, A. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*; CRC Press: Boca Raton, FL, USA, 2018.
13. Simon, D. *Evolutionary Optimization Algorithms*; Wiley: New York, NY, USA, 2013.
14. Ławrynowicz, A. *Genetic Algorithms for Advanced Planning and Scheduling in Supply Networks*; Difin: Warsaw, Poland, 2013.
15. Xu, W.; Jia, H. Research on Storage Location Optimization Based on Genetic Algorithms. *J. Phys. Conf. Series* **2019**, *1213*, 032020. [CrossRef]
16. Hassanat, A.B.A.; Alkafaween, E. On Enhancing Genetic Algorithms Using New Crossovers. *Int. J. Comput. Appl. Technol.* **2017**, *55*. [CrossRef]
17. Hwang, H. An improvement model for vehicle routing problem with time constraint based on genetic algorithm. *Comput. Ind. Eng.* **2002**, *42*, 361–369. [CrossRef]
18. Tan, H.; Lee, L.H.; Zhu, Q.; Ou, K. Heuristic methods for vehicle routing problem with time windows. *Artif. Intell. Eng.* **2001**, *16*, 281–295. [CrossRef]
19. Puljić, K.; Manger, R. Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Math. Commun.* **2013**, *18*, 359–375.
20. Davarzani, H.; Norrman, A. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271.
21. Floyd, R.W. Algorithm 97: Shortest Path. *Commun. ACM* **1962**, *5*. [CrossRef]
22. Bellman, R. On a routing problem. *Q. Appl. Math.* **1958**, *16*, 87–90. [CrossRef]
23. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
24. Kreinovich, V.; Olac, L.F.; Quintana, C. Genetic Algorithms: What Fitness Scaling Is Optimal? *Cybern. Syst.* **2001**, *24*. [CrossRef]
25. Razali, N.M.; Geraghty, J. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. In Proceedings of the World Congress on Engineering WCE 2011, London, UK, 6–8 July 2011.
26. Hassanat, A.E.A. Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information* **2019**, *10*, 390. [CrossRef]
27. Otman, A.; Tajani, C.; Abouchabaka, J. Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem. *arxiv* **2012**, arXiv:1203.3099.
28. Nilsson, C. *Heuristics for the Traveling Salesman Problem*; Technical Report; Linkoping University: Linkoping, Sweden, 2003.
29. Kaabi, J.; Harrath, Y. Permutation rules and genetic algorithm to solve the traveling salesman problem. *Arab. J. Basic Appl. Sci.* **2019**, *26*, 283–291. [CrossRef]
30. Kordos, M.; Łapa, K. Multi-Objective Evolutionary Instance Selection for Regression Tasks. *Entropy* **2018**, *20*, 746. [CrossRef]

31. Kordos, M.; Arnaiz-González, A.; García-Osorio, C. Multi-Objective Evolutionary Instance Selection for Regression Tasks. *Neurocomputing* **2019**, *358*, 309–320. [CrossRef]
32. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
33. Kordos, M. Optimization of Evolutionary Instance Selection. *Lect. Notes Artif. Intell.* **2017**, *10245*, 359–369.
34. Xin, J.; Zhong, J.; Yang, F.; Cui, Y.; Sheng, J. An Improved Genetic Algorithm for Path-Planning of Unmanned Surface Vehicle. *Sensors* **2019**, *19*, 2640. [CrossRef]
35. Contreras-Bolton, C.; Parada, V. Automatic Combination of Operators in a Genetic Algorithm to Solve the Traveling Salesman Problem. *PLoS ONE* **2015**, *26*. [CrossRef]
36. Contreras-Bolton, C.E. Algorithms for Variants of Routing Problems. Ph.D. Thesis, Università di Bologna, Bologna, Italy, 2019.
37. Łapa, K.; Cpałka, K.; Laskowski, Ł.; Cader, A.; Zeng, Z. Evolutionary Algorithm with a Configurable Search Mechanism. *J. Artif. Intell. Soft Comput. Res.* **2020**, *10*, 151–157. [CrossRef]
38. Chehouri, A.; Younes, R.; Perron, J.; Ilinca, A. A Constraint-Handling Technique for Genetic Algorithms using a Violation Factor. *J. Comput. Sci.* **2016**. [CrossRef]
39. Ponsich, A.; Azzaro-Pantel, C.; Domenech, C.; Pibouleau, L. Constraint handling strategies in Genetic Algorithms application to optimal batch plant design. *Chem. Eng. Process. Process. Intensif.* **2008**, *47*, 420–434. [CrossRef]

applied sciences

MDPI

Article

Application of Machine Learning Techniques to Delineate Homogeneous Climate Zones in River Basins of Pakistan for Hydro-Climatic Change Impact Studies

Ammara Nusrat *, Hamza Farooq Gabriel, Sajjad Haider, Shakil Ahmad, Muhammad Shahid and Saad Ahmed Jamal

School of Civil and Environmental Engineering, National University of Sciences and Technology, Islamabad 44000, Pakistan; hamza.gabriel@nice.nust.edu.pk (H.F.G.); sajjadhaider@nice.nust.edu.pk (S.H.); shakilahmad@nice.nust.edu.pk (S.A.); m.shahid@nice.nust.edu.pk (M.S.); sahmed.be16igis@igis.nust.edu.pk (S.A.J.)

* Correspondence: ammara.phd14nice@student.nust.edu.pk; Tel.: +92-346-500-1107

Received: 15 August 2020; Accepted: 28 September 2020; Published: 1 October 2020

Abstract: Climatic data archives, including grid-based remote-sensing and general circulation model (GCM) data, are used to identify future climate change trends. The performances of climate models vary in regions with spatio-temporal climatic heterogeneities because of uncertainties in model equations, anthropogenic forcing or climate variability. Hence, GCMs should be selected from climatically homogeneous zones. This study presents a framework for selecting GCMs and detecting future climate change trends after regionalizing the Indus river sub-basins in three basic steps: (1) regionalization of large river basins, based on spatial climate homogeneities, for four seasons using different machine learning algorithms and daily gridded precipitation data for 1975–2004; (2) selection of GCMs in each homogeneous climate region based on performance to simulate past climate and its temporal distribution pattern; (3) detecting future precipitation change trends using projected data (2006–2099) from the selected model for two future scenarios. The comprehensive framework, subject to some limitations and assumptions, provides divisional boundaries for the climatic zones in the study area, suitable GCMs for climate change impact projections for adaptation studies and spatially mapped precipitation change trend projections for four seasons. Thus, the importance of machine learning techniques for different types of analyses and managing long-term data is highlighted.

Keywords: climate zone; climate change impact; Jhelum River Basin; Chenab River Basin

1. Introduction

Climate change has emerged as a major driving force behind Pakistan's flooding over the past several decades, mainly by affecting snow melting and interfering with summer monsoon patterns [1]. Previous studies have also shown a growing interest in an accurate and calculated evaluation of climate change that can provide a rational system to adapt to a rapidly changing environment. Climate change impact analyses use long records of climate data from numerous gauging sites to estimate the historical and projected climate change trends. There is a growing institutional and social need to predict the potential impacts of climate change and the damage due to floods and droughts owing to spatio-temporal variability in the climate, specifically the rainfall. Previous studies have established that spatio-temporal dynamics in atmospheric covariates are strongly linked to the precipitation characteristics on a regional scale. The atmospheric circulation covariates are important in defining the precipitation patterns across the region [2]. Different atmospheric oscillations are responsible for this variability, which has been characterized by various climate models/simulations, including several parameters related to atmospheric oscillation moisture and wind fluxes [3]. The driver of seasonal rainfall in southeast Asia is the Monsoon circulation [4] and increases in the probability and severity of extreme events (e.g., droughts, heatwaves and floods) have been documented in several studies [5,6]. The spatio-temporal variability of the hydrological cycle parameters affects the social and environmental sustainability [7]. Therefore, the patterns and projected trends of precipitation, which is the most important variable of the hydrological cycle varies spatially as well as temporally, must be studied.

Simulation results from general circulation models (GCMs) are used to study the implications of hydro-climatic change and are considered the most comprehensive and valid instruments, with the ability to reproduce climate variables in chronological order as well as yield future projections using scenarios [8–11] recommended by the intergovernmental panel on climate change (IPCC). Numerous outputs from GCMs are available to gain insight into climate processes and the impact of various scenarios on these processes [12,13]. The question arises as to how to use these climate simulations to obtain meteorological inputs for impact models to characterize all the uncertainties involved in the models. Previous studies have reported attempts to investigate the suitability of climate models when simulating different climate parameters, such as precipitation [14]. spatio-temporal distribution by the GCM is an important aspect that must be considered for the selection of the model [15]. Further discussion regarding the selection of appropriate GCMs is presented in Section 3.2.

Climate model simulations should have adequate precision to assimilate crucial information in the observational data, which forms the basis to forecast and more efficiently predict climatic conditions. The sources of uncertainty in climate simulations and predictions are (1) model equations parameterization and initial conditions associated with chaotic systems [16]; (2) the variability in the predicted scenario caused by anthropogenic forcing, which is unknown in the future; and (3) climate variability [17]. To account for these uncertainties, multiple GCM assessment studies analyze model efficiency at simulating the dynamic and thermodynamic variables and the environmental variables related to atmospheric chemistry [4,18,19]. The common trend among the climate research community, however, is the statistical performance evaluation of GCM using climate output variables from its simulations [7,19–23]. These climate variables are averaged temporally and spatially on various scales for the assessment of the GCM's ability to either simulate historical data or to present the range of climate output variables to detect a plausible future [22,24].The majority of assessment studies of climate change used the following two approaches to evaluate the climate model skill to represent past observational climate data:

1. comparing the spatio-temporal average of historical/baseline observational and GCM data of different climate variables [22,25,26], using several performance indicators [27–29]; however, changes in the spatio-temporal average variable values may not represent the change in extreme values [30].
2. comparing the observational and GCM data at each grid point [21,31,32] and then selecting the GCM in the entire study area using different filtering approaches, such as clustering hierarchy [33,34], Bayesian weighting [35], weighted skill score [15,36] and spectral analysis [37]; however, such selection processes do not cater to the area-specific efficiencies of the climate models because this efficiency varies amongst models and regions [4,27]. Several studies [21,31,32] have shown that the evaluated GCMs only account for some grid points over the entire study region. However, these studies have overlooked the ability of GCM to reproduce spatial climatic patterns. The drivers of this spatial variability are atmospheric circulation patterns that depend on the geographical location of the area. The capacity of the GCMs to reproduce the patterns of climate parameters spatially, in the baseline/historical period is, therefore, imperative [7,21]. The GCM or ensemble of GCMs selected, via the aforementioned comparative analysis of spatio-temporal averages and filtering techniques, may be appropriate for simulating the spatial climatic patterns for some percentage of the entire study area but not all, thus enhancing the uncertainties in projecting the climate. To narrow this uncertainty in projecting the climate variables (precipitation) trends, we propose a different method of the selection of GCMs by first

identifying the climatologically homogeneous zones and then selecting the GCM in each zone through past performance assessment. These homogeneous sub-regions are based on similar spatial and temporal climate patterns, as depicted by the highly resolved observation data. Following the philosophy of McSweeny et al. [4] for the regional suitability of the GCMs, we selected the GCMs in every homogeneous precipitation region and validated our selection through the comparative analysis with Simple Composite Method (SCM). The SCM is broadly used for generating the multi-model ensemble that is, to obtain the equally weighted mean of all the ensemble members data at each grid point [7].

With the support of machine learning techniques, long records of climate data, from numerous gauging sites and web sources, can be easily analyzed and used to determine historical and projected trends of climate change. In the present study, the modules using machine learning techniques were developed throughout the various steps of the framework. The framework comprises five important steps: (1) developing homogeneous precipitation zones using highly resolved observation data; (2) the selection of the best suitable GCM for each zone based on the estimated correlation coefficient using the GCM data and highly resolved observed data in every homogeneous climatic region; (3) comparison and validation of the selected GCM; (4) sampling the daily precipitation data for the projected period (2006–2099); based on the forcing scenarios and (5) performing future precipitation trend detections. The outcomes of this framework are the estimated divisional boundaries of the climate zones, suitable GCMs for the climate impact studies and seasonal rainfall trend projections. It is necessary to mention that the results obtained using the present framework are subject to specific assumptions: (1) daily and 0.25° temporal and spatial scales of the data have been used for climate zoning (2) seasonal and 0.25°, temporal and spatial scales of the data have been used for the climate projection trends (3) uncertainty of the results depends on the uncertainty involved in the climate variable outputs from the GCMs. It is very important to understand the limitations and uncertainties involved in the simulation outputs of different GCMs. The present study does not aim to quantify the uncertainties present in the GCMs' simulations but only uses the GCMs' outputs for their selection in each climate zone and trend projections. Nonetheless, the framework can be replicated, using other versions of ensembles of GMCs' with finer resolution and accuracy. The scientific data associated with the present framework can be shared on request.

This paper is structured as follows—Section 2 describes the study area, stations and data details; Section 3 provides a step-by-step presentation of the methodology; Section 4 presents the results and discussion; and Section 5 presents the conclusions.

2. Study Area and Data

2.1. Study Area

The basin areas of the Jhelum and Chenab Rivers are 33,330 and 67,515 km^2, respectively. The elevations in the two basins vary between 146 and 6915 m. Figure 1 shows a topographic map with APHRODITE grid points of the basins of the Jhelum and Chenab river in Pakistan. The Southwest Monsoon triggers approximately 40% of the annual precipitation, whereas the remaining ~60% of precipitation is due to Westerly aggravations [24]. Figure 2 presents the average annual seasonal precipitation in various parts of the two basins. The spatially and temporally variable climatic conditions, atmospheric circulation, advected moisture and topography, with high elevations in the north and flat plains in the south, demand a flexible framework for the evaluation of the climate change effects and future predictions. Diminishing water resources due to high hydro-climatic variability and extreme climate events in Pakistan are some of the factors that inspired the research community to investigate sustainable solutions after analyzing future needs and availability.

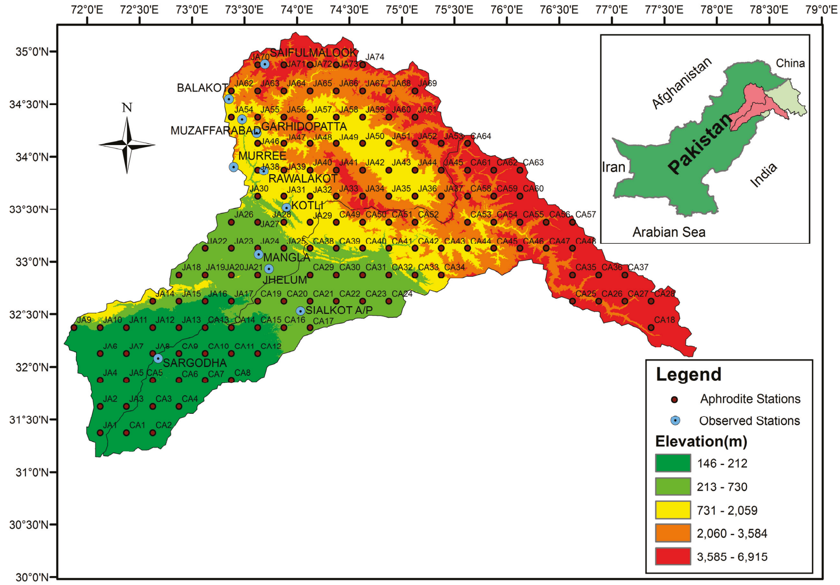

72°0'E
72°30'E
73°0'E
73°30'E
74°0'E
74°30'E
75°0'E
75°30'E
76°0'E
76°30'E
77°0'E
77°30'E
78°0'E
78°30'E
79°0'E
35°0'N
34°30'N
34°0'N
33°30'N
33°0'N
32°30'N
32°0'N
31°30'N
31°0'N
30°30'N
30°0'N
N
SAIFULMALOOK
BALAKOT
MUZAFFARABAD
GARHIDOPATTA
MURREE
RAWALAKOT
KOTLI
MANGLA
JHELUM
SIALKOT A/P
SARGODHA
JA70 JA71 JA72 JA73 JA74
JA62 JA63 JA64 JA65 JA66 JA67 JA68 JA69
JA54 JA55 JA56 JA57 JA58 JA59 JA60 JA61
JA46 JA47 JA48 JA49 JA50 JA51 JA52 JA53 CA64
JA38 JA39 JA40 JA41 JA42 JA43 JA44 JA45 CA61 CA62 CA63
JA30 JA31 JA32 JA33 JA34 JA35 JA36 JA37 CA58 CA59 CA60
JA26 JA27 JA28 JA29 CA49 CA50 CA51 CA52 CA53 CA54 CA55 CA56 CA57
JA22 JA23 JA24 JA25 CA38 CA39 CA40 CA41 CA42 CA43 CA44 CA45 CA46 CA47 CA48
JA18 JA19 JA20 JA21 CA29 CA30 CA31 CA32 CA33 CA34 CA35 CA36 CA37
JA14 JA15 JA16 JA17 CA19 CA20 CA21 CA22 CA23 CA24 CA25 CA26 CA27 CA28
JA9 JA10 JA11 JA12 JA13 CA13 CA14 CA15 CA16 CA17 CA18
JA6 JA7 JA8 CA9 CA10 CA11 CA12
JA4 JA5 CA5 CA6 CA7 CA8
JA2 JA3 CA3 CA4
JA1 CA1 CA2
Afghanistan
China
Iran
Pakistan
India
Arabian Sea
Legend
● Aphrodite Stations
● Observed Stations
Elevation(m)
🟩 146 - 212
🟩 213 - 730
🟨 731 - 2,059
🟧 2,060 - 3,584
🟥 3,585 - 6,915

Figure 1. Topography of the study area with the grid points/stations and observed gauging station used in the study.

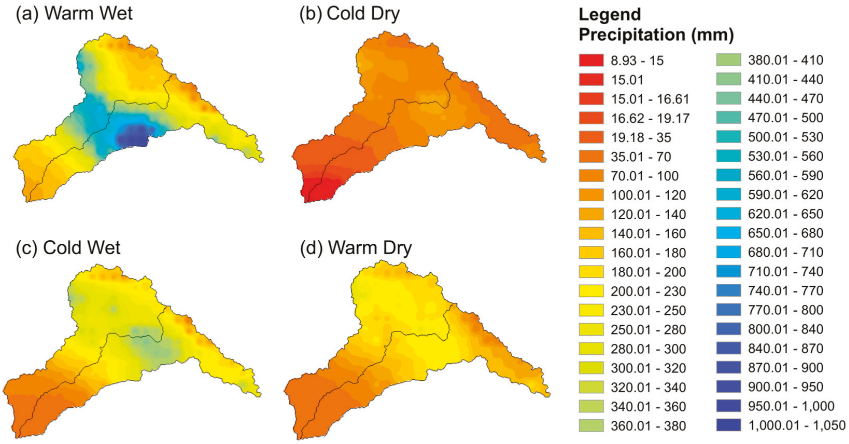


Figure 2. The distribution of average total seasonal precipitation based on APHRODITE (1970–2004) for (**a**) warm-wet (**b**) cold-dry (**c**) cold-wet and (**d**) warm-dry seasons.

2.2. APHRODITE Data

To delineate the climate zones in the study area, a dense network of data stations was required. There has been an increasing trend of using easily available gridded precipitation datasets for hydrologic and climatic assessments [38–40]. Previous studies have identified the performance of the APHRODITE dataset (version V1101) as the best-gridded product over the high mountainous regions of Asia [41].

To justify the use of the gridded dataset of APHRODITE, we performed the comparative analysis of the APHRODITE dataset and European Reanalysis gridded dataset (ERA5) [42] and Global Metrological forcing dataset for land surface Modeling (GMFD) [43] with the observed dataset and monthly temporal scale, at 11 gauging stations at different altitudes, as shown in Figure 1. ERA5 is also popular for its accuracy. The results of the comparative analysis have been shown in Table 1. The Pearson correlation coefficient and Kolmogorov Simirnov (KS) test (the method is discussed in Section 3.3.2) were used to compare the monthly precipitation datasets. The results tend to show agreement with the APHRODITE dataset with higher correlation coefficients at all the stations. For the KS test, *p*-values greater than 0.05 have been shaded, depicting the null hypothesis of similar probability distribution is not rejected. The results of comparative analysis suggests that the APHRODITE datasets have higher correlation with the observed data that is, more than 0.8 in nearly all the observed gauging station as shown in Table 1, so the APHRODITE dataset [40] of 0.25° × 0.25° spatial resolution for the period of 1970–2005 has been used at a daily temporal scale, for the delineation of climate zones. Nevertheless, the regionalization framework is equally applicable to the other datasets having comparatively higher resolution and accuracy.

Table 1. Comparative analysis of APHRODITE and ERA5 monthly dataset. Pearson correlation coefficients and Kolmogorov Smirnov Test results (KS Test). (The shaded *p*-Values are >0.05, depicting the null hypothesis of similar distribution, is not rejected).

Gauging Stations	Time Period	Correlation Coefficient				*p*-Value, KS Test		
		Elevation (m)	APHRODITE	ERA5	GMFD	APHRODITE	ERA5	GMFD
Astore	1979–2005	2546	0.90	0.65	0.08	0.0001	0.000	0.001
Balakot	1979–2005	1088	0.91	0.77	0.17	0.018	0.069	0.000
GarhiDopatta	1979–2005	819	0.90	0.71	0.25	0.0001	0.336	0.000
Jhelum	1979–2005	234	0.99	0.79	0.31	0.99	0.000	0.238
Kotli	1979–2005	608	0.91	0.83	0.26	0.29	0.568	0.079
Mangla	1996–2005	605	0.96	0.77	0.40	0.74	0.519	0.060
Muzaffarabad	1979–2005	737	0.92	0.71	0.28	0.92	0.248	0.000
Rawalakot	2003–2005	1638	0.95	0.75	0.10	0.11	0.199	0.024
Saifulmaluk	1996–2005	3224	0.37	0.52	0.13	0.0006	0.00	0.069
Sargodha	1979–2005	190	0.78	0.75	0.26	0.0002	0.000	0.002
Sialkot	1979–2005	256	0.91	0.79	0.35	0.29	0.014	0.201

Figure 2 presents the total seasonal average precipitation sampled from APHRODITE. The daily gridded precipitation long term data from 1951 and onwards, is provided by the APHRODITE product (V1101). It provides the data at the continental scale, including a dense rain gauge data network of Asia, comprising the Middle East, South and Southeast Asia with valid stations of 5000 to 12,000 [40].

2.3. NEX-GDDP-GCMs-CMIP5 Data

For future climate change analysis, this study used newly developed NASA Earth Exchange Global Daily Downscaled Projections (NEX-GDDP) dataset. The experiments included in Coupled Model Intercomparison Project 5 (CMIP5) were devised to address the research interrogations in AR4 of the IPCC [44]. The original resolution of most GCMs in the CMIP5 is >100 km, where such coarse resolution cannot provide fine-scale information for impact studies and decision-making at a local scale. The NEX-GDDP dataset is a collection of 21 GCMs. These datasets are bias-corrected and downscaled to a finer resolution (0.25° × 0.25°) using the bias-corrected spatial disaggregation (BCSD) method [45]. The Global Meteorological Forcing Dataset (GMFD), developed by Princeton University, provides the gridded observed climate data and the NEX-GDDP after bias correction and downscaling [46,47]. Table S1 lists the 21 GCMs in the NEX-GDDP, along with information on the research centers that produce the GCMs.

NEX-GDDP data consists of daily precipitation and minimum and maximum temperatures from historical periods (1950–2006) and future projections (2006–2099). The future projections are available for two representative concentration pathways (RCPs), which are global greenhouse gas emissions scenarios. These scenarios were employed by the IPCC fifth assessment report (AR5) [44].

The NEX-GDDP dataset is publicly available. The comprehensive details to use these scenarios dataset for climate change impact evaluations and adaptations have been provided by IPCC [48]. The CMIP5 data provides multi-model datasets of climate variability, which was used to develop AR5 [44].

3. Methodology

Machine learning algorithms are becoming more popular owing to their ability to identify the patterns and variances in large datasets composed of multivariate atmospheric covariates, location parameters, meteo-climatic information, tele-connection indices and attributes that influence precipitation [2,49,50] or the precipitation and temperature statistics [51,52]. In this study, we used the Python module scikit-learn [53], which integrates a broad range of machine learning algorithms for supervised as well as unsupervised learning. As aforementioned, this framework consists of five steps: delineation of homogeneous climate zones; selection of GCMs; validation of selected GCMs; selection of forcing Scenarios to be used for the predictions of the climate change; and climate change projections and trend detection. Figure 3 presents the methodology flow diagram adopted in this study. Further details of each step are discussed in the subsequent sections.

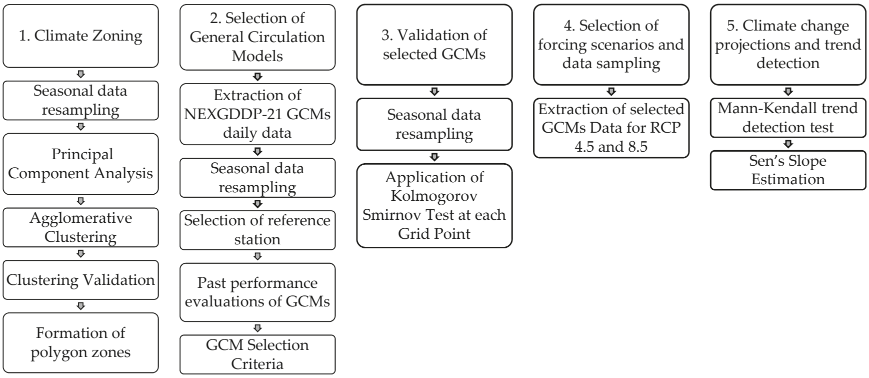

1. Climate Zoning
Seasonal data resampling
Principal Component Analysis
Agglomerative Clustering
Clustering Validation
Formation of polygon zones
2. Selection of General Circulation Models
Extraction of NEXGDDP-21 GCMs daily data
Seasonal data resampling
Selection of reference station
Past performance evaluations of GCMs
GCM Selection Criteria
3. Validation of selected GCMs
Seasonal data resampling
Application of Kolmogorov Smirnov Test at each Grid Point
4. Selection of forcing scenarios and data sampling
Extraction of selected GCMs Data for RCP 4.5 and 8.5
5. Climate change projections and trend detection
Mann-Kendall trend detection test
Sen's Slope Estimation

Figure 3. Flow chart of the methodology employed in this study.

3.1. Climate Zoning

Climate zoning/regionalization is the most important step in this framework and the basis for the subsequent steps. For regionalization, solid evaluations of different precipitation attributes require accessibility to long records of chronicled estimations at various stations inside an area [2]. For climate zoning, 35 years' (2006–2075) daily precipitation data of APHRODITE were used. These data were resampled seasonally to develop the climate zones for each of the seasons (defined in the subsequent sections). Regionalization was performed by grouping the rain gauges with homogeneous precipitation statistics in an area [54]. This step enables partitioning of the entire area into several climate zones for every season.

The regionalization of the precipitation statistics has numerous applications in various water resource management fields, such as agriculture practices, spatial and temporal rainfall patterns, hydrological analysis, extreme event forecasting [55] and watershed management. There are several methods available to delineate climate regions, for example, subjective and objective partitioning, geographical convenience and multivariate analysis [54,56,57]. The arbitrary and slightly misleading demarcation approach for a region, which is based on administrative boundaries and physical and geographical groupings, is referred to as the geographical convenience method. The subjective partitioning method is based on homogeneous statistical characteristics of the rainfall and previous knowledge of the region. Objective partitioning demarcates a region by grouping the sites of similar climates. Principal component analysis (PCA), clustering techniques and correlation analysis are examples of multivariate analysis techniques, which are widely used for climate zone delineation [51,58,59]. These analysis techniques require the development of a large matrix that defines the characteristics of the climate in the region. We adopted the method of PCA and agglomerative hierarchical clustering (AHC) to group the sites of homogeneous precipitation. These groups of stations were validated using different cluster validity indices.

3.1.1. Seasonal Data Resampling

The daily precipitation data in the APHRODITE time series, at 138 Grid stations in the study area from 1975–2005, were resampled as hydrological seasons: warm-wet ("July, August and September"), cold-dry (October, November and December), cold-wet (January, February and March) and warm-dry (April, May and June). The daily precipitation data of 35 years were resampled in a seasonal cycle using simple Python code.

3.1.2. Principal Component Analysis (PCA)

The benefits associated with the PCA are as follows—(1) identification of spatially coherent variations that can improve the signals, where the leading components represent the maximum variance in the patterns; (2) explanation of the covariance between the variables at various places; and (3) reduction of dimensionality, thereby providing resources to describe the variability in large spatially dimensional datasets with a reduced number of principal components [60]. PCA can present all the data in a smaller matrix and attempts to retain as much information about the data as possible. We used the dimensions from 138 APHRODITE grid point datasets for PCA. The aim was to project the data onto different orthogonal axes known as principal components. A linear transformation is performed using a symmetric covariance matrix, developed from the dataset, into the principal orthogonal components. The direction of principal components is represented by eigenvectors and the eigenvalues represent the magnitude of the stretch of the axis. This gives the direction (eigenvector) and magnitude (eigenvalue) of the spread of the dataset. The leading principal component, which presents the maximum spread of the dataset, is used for further analysis. Several principal components were selected for use in hierarchical clustering using the scree plot, which shows the variance explained in percentage by each of the principal components. Each principal component has component scores for every station, based on the eigenvectors and eigenvalues. These component scores depict the climate change pattern at the respective grid point/station and can be treated as metrological parameters that are stochastically independent of each other [61]. This analysis has been performed using the PCA function of scikit-learn [53].

3.1.3. Agglomerative Hierarchical Clustering

The purpose of this step is to obtain the groups of sites/ stations with similar climate change patterns depicted by the component scores of the leading principal components obtained in the previous step. This is achieved using different clustering algorithms. Different behaviors in the clustering algorithm can be expected based on the data features, dimensions and input variable values. Traditionally, different clustering algorithms are used in previous studies; there is extensive literature regarding these clustering techniques [51,61–63]. Recently, the agglomerative method of hierarchical clustering [61,64] has received increased attention in climate literature. In this method, smaller clusters are developed according to a bottom-up approach, followed by a sequential combination with larger clusters depending on the Euclidian distance [65] between the clusters.

In the algorithm, each evaluation (climate change pattern) was allocated to its self-cluster. Subsequently, the iterations in the algorithm were identified and joined with the closest cluster. This agglomeration continued until the formation of one cluster. A tree-like structure (dendrogram) of similarity was formed because of the hierarchical clustering, which provides meaningful information regarding the correlations/distance among different clusters. The validity of the number of clusters (CN) was determined prior to the identification of the maximum valid Euclidian distance. The agglomerative clustering algorithm of scikit-learn [53] was utilized for grouping the sites for each season, in this study.

3.1.4. Optimal Clustering and Climate zone Formation

In this step, the optimal CN of stations for each season is identified. Optimal clustering validity methods must be employed for the real grouping of datasets. Comprehensive studies have been performed for the comparative analyses of the cluster validity indices. The silhouette index (*S*) [66], S Dbw index (S Dbw) [67] and Calinski–Harabasz index (*CH*) [68] have been used in this study following the recommendations on these comparative evaluations [67,69].

Using the average of the three values identified through the aforementioned validity indices, the optimal CN was identified. The numbering and identification of stations in each cluster were performed via the truncation of the dendrogram. The truncation (cut-off) bar is kept at the valid maximum Euclidian distance corresponding to the validated value of CN [20]. This was performed using a supervised learning algorithm in scikit-learn [53].

Silhouette Score

The silhouette score method [66] is employed in the analysis to obtain the optimum CN. The silhouette score is the mean of the distance between clusters. In this study, the silhouette score yielded a maximum of 14 clusters for climate zoning for all seasons. The CN corresponding to the maximum silhouette score is the basis for the decision. The Silhouette score (*S*) is obtained as

$$S = \frac{1}{CN}\sum_{i}\frac{1}{n_i}\sum_{r\in C_i}\frac{b(r)-a(r)}{\max[b(r),a(r)]} \tag{1}$$

and

$$a(r) = \frac{1}{n_i-1}\sum_{s\in C_i, s\neq r} d(r,s), b(r) = min_{j,j\neq i}\left[\frac{1}{n_j}\sum_{y\in C_j} d(r,s)]\right] \tag{2}$$

where the number of clusters is denoted by CN; C_i represents the *i*th cluster; n_i represents the number of objects in C_i; c_i denotes the center of C_i; and *d*(r,s) is the distance between r and s [66].

S Dbw Validity Index

S Dbw considers the inter-cluster density to calculate inter-cluster separation [67]. One of the densities of each cluster pair must be greater than the midpoint density for the cluster centers. This index is the summation of the scatter in the clusters and the density between the clusters. Previous studies have investigated the validation properties of different cluster validity indices using synthetic experimental data [70]. They conclude that S Dbw performed best concerning the aspects of noise, monotonicity, density, skewed distribution and sub-clusters. Equations (3)–(5) can be used to calculate the density between clusters, the scatter in the clusters and the S Dbw, respectively:

$$Dens\_bw(CN) = \frac{1}{CN(CN-1)}\sum_{i=1}^{CN}\left[\sum_{j=1,i\neq j}^{CN}\frac{\sum_{x_i\in C_i\cup\ C_j} f\left(x_i,u_{ij}\right)}{\max\left\{\sum_{x\epsilon C_i} f(x,c_i),\sum_{x\epsilon C_j} f\left(x,c_j\right)\right\}}\right] \tag{3}$$

$$Scat(CN) = \frac{1}{CN}\sum_{i=1}^{CN}\frac{\|\sigma(C_i)\|}{\|\sigma(D)\|}, Dis(CN) = \frac{max_{i,j}\, d\left(c_i,c_j\right)}{min_{i,j}\, d\left(c_i,c_j\right)}\sum_{i=1}^{CN}\left(\sum_{j=1}^{CN} d\left(c_i,c_j\right)\right)^{-1} \tag{4}$$

$$S\_Dbw = Scat(CN)\ +\ Dens\_bw(CN), \tag{5}$$

where D is the dataset; CN is the number of clusters; $\sigma(C_i)$ is the variance vector of C_i, $\sigma(D)$ is the variance in the dataset, (c_i, c_j) is the distance between c_i and c_j and $\|x\| = \left(x^T x\right)^{1/2}$.

Calinski–Harabasz Index

This cluster validation scheme depends on the average of the cluster sum of squares and the average of the sum of squares within clusters and is known as CH [68]. It is obtained by formula as shown in Equation (6)

$$CH = \frac{\sum_i n_i d^2(c_i, c)/(CN-1)}{\sum_i \sum_{x\in C_i} d^2(x, c_i)/(n-CN)}. \tag{6}$$

Here, n represents the number of points/objects in D, c denotes the center of D and d (x, c_i) denotes the distance between x and c_i.

3.1.5. Climate Zone Polygon Formation

The realistic partition of climatic stations for coherent climate regions is a result of this method. The delineation of these regions depends on the similarity among the statistical climate characteristics of all the stations present in a region.

The clusters of stations for every season were plotted using the ArcGIS (Environmental Systems Research Institute, CA, USA) interface. After the identification of a group of stations in the different clusters, approximate cluster boundaries were drawn on the maps using the Arc-map tool to enable a clearer presentation.

3.2. GCM Selection

The criteria used to select the GCMs are the availability of the latest generation of GCMs, good spatial resolution, past performance of GCM to replicate the historical data and the representativeness of the GCM for a wide range of climatic variable (precipitation) projections [7]. The most commonly adopted criteria are the assessment and selection of GCMs based on their capability to simulate the historical and present climate, which have been adopted by numerous studies [8–10]. An efficient and sensible selection of GCMs is required to generate reliable and diverse meteorological inputs for the impact models. Most previous studies show that a past performance assessment is among the most effective methods for the selection of GCMs, as the GCMs thus selected can be better predictors of future climatic conditions [31].

Several selection methods have been reported in the literature. Aghakhani et al. [71] extracted GCM data at four points in the vicinity of observation stations and performed a comparative analysis between the GCM and the observed data. They used the averages of the time-series data at each station for their analysis. Xuan et al. [72] selected GCMs using an average of the climate parameters for the entire study area in the Zhejiang Province of Southeast China. Najeebullah et al. [32] ranked the GCMs at every grid point after re-gridding the GCM data at the same spatial resolution, comparing the data with the gridded data product of the Asian Precipitation - Highly-Resolved Observational Data Integration Towards Evaluation (APHRODITE) for GCM performance evaluations in all of Pakistan. When comparing the best GCM with the highly ranked GCM at each grid station, they found a significant difference between the two with respect to precipitation. Maxino et al. [15] analyzed the climate models from the Fourth Assessment Report (AR4) of the United Nations Intergovernmental Panel on Climate Change (IPCC) for two regions in the Murray Darling Basin, divided based on rainfall classification. For different climate variables, they demonstrated that the models that are flawed in one region may be better for another region; model selection should be area specific. They calculated the skill scores associated with the different climate models in the study region. Lutz et al. [22] averaged the climate data over 2.5° × 2.5° grid cells in three river basins (i.e., the Indus, Brahmaputra and the Ganges) and presented their model with the selected GCMs. Latif et al. [73] also suggested certain

Coupled Modelled Intercomparison Project 5 (CMIP5) GCMs after evaluating their performance using seasonal rainfall spatial correlations in the Indo-Pak region. Ahmed et al. [7] evaluated the spatial accuracy of the CMIP5 GCMs using different spatial metrics for all of Pakistan. They performed the analysis after re-gridding the GCM data into a 2° × 2° grid size. Srinivasa et al. [31] ranked the GCMs for the maximum and minimum temperatures across India. This framework comprised the evaluation of GCM suitability at every grid point and then ranking the GCMs using compromised programming.

In the present study, after defining the clusters of stations with homogeneous precipitation, one representative station was selected in every climate zone, using quota sampling [61]. The selection of the representative station was done based on the average climate signal climate signals (Component Scores) in a respective climate zone. The comparative analysis of GCMs data and APHRODITE data at the representative station of each climate zone was accomplished using the coefficient of determination (R^2).

A reduced correlation existed within the daily outputs of the GCMs and the in situ data. However, better correlations were obtained when seasonal data in the GCMs were compared with the seasonal observation data. Therefore, using the seasonal data, the correlations between the GCM data and observed gridded data were evaluated at each representative station in the respective climate zone.

3.3. Validation of Selected GCMs

This is the important step of comparative analysis of the climatic data generated through the present selection method of the GCMs and the contemporary method of sampling through simple mean based multi-model Ensemble (MME) data. The KS test [74], a nonparametric test, was used to determine the validity of the selected GCM by detecting the changes in the distribution of the two datasets (dataset generated through present selection method and simple mean based MME data) with the APHRODITE. For every grid station, the observational precipitation data (APHRODITE), selected GCM simulated data of precipitation and multi-model ensemble(MME) mean daily precipitation data of all the 21 GCMs of NASA Earth Exchange Global Daily Downscaled Projections (NEX-GDDP) CMIP5 for the same baseline period from 1970–2005 were used in this test.

3.3.1. Seasonal Data Sampling

The daily precipitation data of the selected GCM, corresponding to each homogeneous climate zone, were seasonally resampled for every grid station (i.e., the grid station within each grid cell). The multi-model ensemble mean data were generated by resampling the data as the average of the GCM daily precipitation data at every grid station.

3.3.2. Kolmogorov–Smirnov Test

In this study, the KS test stats were derived as the highest vertical difference between the two cumulative distribution functions (CDFs) of a time series data. For CDF of APHRODITE data $ø_n(x)$ and the CDF of the selected GCM or MME mean at a grid station $ø\ (x)$, the KS test statistics were obtained as given in Equations (7) and (8).

$$S_n = sup_x \left| o_n(x) - o(x) \right|, \tag{7}$$

where

$$o_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{x_i \le x}. \tag{8}$$

Here, sup_x is the upper bound of the set of distances. The indicator function is represented by $I_{x_i \le x}$ (If $x_i \ \le \ x$, $I_{x_i \le x}$ equal to 1 or if $x_i \ > \ x \ I_{x_i \le x}$ equal to 0). The null hypothesis of a similar distribution is rejected if $S_n \ge 0.05$ significance level. The KS test was applied at every grid station using scikit-learn.

3.4. Selection of Forcing Scenarios

Four climate forcing scenarios are commonly used in climate research for climate simulation studies on both long term and short term scales. There are four common Representative Concentration Pathways (RCPs)—RCP 2.6 is a mitigation scenario; RCP 4.5 and RCP 6.0 are scenarios of medium stabilization and RCP 8.5 is a high baseline emissions scenario [75]. We did not include RCP 2.6 for the ensemble of climate models, as it is not considered a realistic or practical scenario necessary to promote adaptation planning. These rest of the three scenarios represent the complete extent of radiative forcing. In this study, we used the RCP 4.5 and RCP 8.5. The same framework applies to other forcing scenarios [22].

Extraction of Selected GCMs Data for RCP 4.5 and 8.5

The daily precipitation data for the selected GCM in the specific climate zone for the projected period from 2006–2099, as well as forcing scenarios RCP 4.5 and 8.5, were sampled for all seasons at every grid station.

3.5. Climate Change Projections and Trend Detection

The highly correlated GCMs, which can replicate the gridded dataset of seasonal precipitation for the period from 1970–2005, were selected in each climate zone for use in the future predictions of precipitation change trends in RCP 4.5 and RCP 8.5. The precipitation change trends were detected using the Mann–Kendall (MK) test during the projected period (2006–2099). The trend was quantified by employing Sen's slope estimator.

3.5.1. Mann-Kendall Test (MK Test)

The MK test [76] was used to evaluate the statistically significant annual seasonal trends in the long term precipitation data. The "no trend" in precipitation with time is assumed in the null hypothesis (H_o) of the test and vice versa for the alternative hypothesis (H_a). The Equations (9)–(12) show the test stats T of the MK test.

$$T = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sig\left(D_j - D_i\right) \tag{9}$$

$$sgn\left(D_j - D_i\right) = \begin{cases} +1 & if\ \left(D_j - D_i\right) > 0 \\ 0 & if\ \left(D_j - D_i\right) = 0 \\ -1 & if\ \left(D_j - D_i\right) < 0 \end{cases} \tag{10}$$

$$\sigma(T) = \frac{1}{18}\left[n(n-1)(2n+5) - \sum_{p=1}^{q} t_p\left(t_p - 1\right)\left(2t_p + 5\right)\right] \tag{11}$$

$$Z = \begin{cases} \frac{T-1}{\sqrt{\sigma\ (T)}} & if\ T > 0 \\ 0 & if\ T = 0\ , \\ \frac{T-1}{\sqrt{\sigma\ (T)}} & if\ T < 0 \end{cases} \tag{12}$$

where D_i and D_j are the consecutive time-series observations, which are organized chronologically for the length of data (n); t_p represents the data points number for the pth value in a tied group; and q denotes the total number of tied groups σ is the variance. An upward time series trend is represented by a positive Z value and vice versa for the negative trend. If $|Z| > Z_{1-\alpha/2}$, the null hypothesis (H_o) is rejected which is indicative of a statistically significant trend. The critical value of $Z_{1-\alpha/2}$ is assumed at the *p*-value of 0.05.

3.5.2. Sen's Slope Evaluation

The Python module was applied to obtain the slope of the trend, if exists, according to Sen's method [77]. According to the method, linear slope sets can be estimated as shown in Equation (13).

$$T_i = \frac{D_j - D_k}{j-k} \text{ for } (1 \leq i < j \leq \text{n}), \tag{13}$$

where Ti is the slope, D_j and D_k are the variables at *j* and *k* time steps, respectively and n represents the data points number. Sen's slope is estimated as the median of all slopes.

4. Results and Discussion

Taking the spatiotemporal average [22,25,73] of the climate dataset at all the grid points of the area at different temporal scales (daily, seasonal or annual) or using various spatial metrics on the results of the individual grid points [21,31,32] are the conventional methods used for the selection of GCMs. However, these methods do not address the spatial coherence of the climate variability patterns in the study area. This poses uncertainty in using these conventional methods. Several studies support the association between atmospheric covariates and precipitation. In GCMs, these atmospheric covariates are the input variables. To address the uncertainty arising from the input variables of the GCMs, it is imperative to assess the GCM performance in reproducing the regional climate pattern. These results show that the climate variability pattern of the study area is spatially heterogeneous in all seasons and a single or an ensemble of GCMs may not represent this spatial climatic heterogeneity. Therefore, we classified/ regionalized the large study in several climate zones founded on the climate variability patterns similarity and the selection of a GCM should be done separately in each of the homogeneous climate zones.

This section presents the detailed results along with the discussion. The results have been visualized using ArcGIS. This software is based on the integration of different machine learning techniques. Section 4.1 describes the results of the analyses performed for the development of climate zones. Section 4.2 shows the results of the GCM selection procedure and Section 4.3 is related to the validation of the method used for the GCM selection. Section 4.4 described the results of the MK test of trend detection and Sen's slope estimation. These tests determined the precipitation change trend projections for the period of 2006 to 2099.

4.1. Climate Zones

The Principal Component Analysis (PCA) performed as the first step in climate zoning provided the details of the climate change pattern in the entire study area. A statistically significant climate pattern heterogeneity was observed in the region when PCA was performed on a seasonal basis.

4.1.1. Principal Component Analysis

The application of the Principal Component Analysis (PCA) was performed on the daily precipitation series data from APHRODITE on a network of 138 stations distributed across the Jhelum and Chenab river basins. This first led to 20 primary and physically significant principle components (PCs), which collectively explained approximately 95% of the total variability in the precipitation throughout the study area. These were retained for the cluster analysis. Scree Plot is shown in Figure 4. The PCA algorithm produced the scree plots, allowing us to select the number of PCs for the cluster analysis. For the warm-wet season, 20 PCs present 95% of the data variance from the station network (Figure 4a). For the cold-dry season, 20 PCs present 94% of the variance in the data from the station network (Figure 4b). For the cold-wet season, 20 PCs explained 95% of the variance in the data from the station network (Figure 4c). For the warm-dry season, 20 PCs explained 95% of the variance in the data from the station network (Figure 4d). The spatial distribution of the component scores is shown in Figure 5, for the first two PCs, explaining the highest percentage of variance in the data.

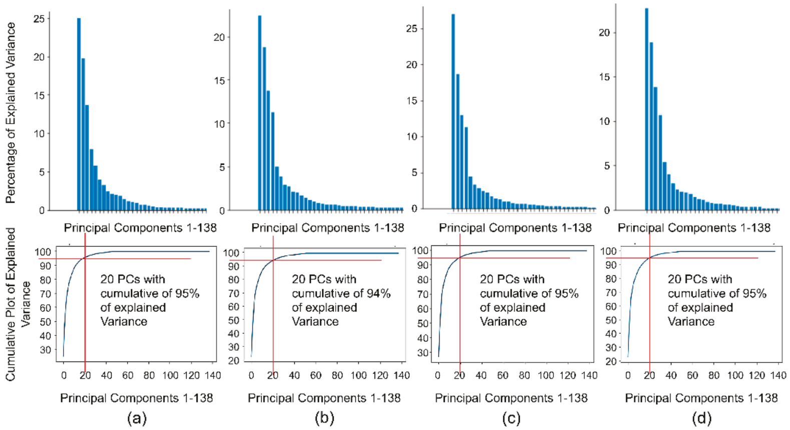


Figure 4. Scree and cumulative plots showing the percentage of explained variance by each principal component (PC) in the (**a**) warm-wet, (**b**) cold-dry, (**c**) cold-wet and (**d**) cold-dry seasons. The red line shows the explained variance corresponding to 20 principal components.

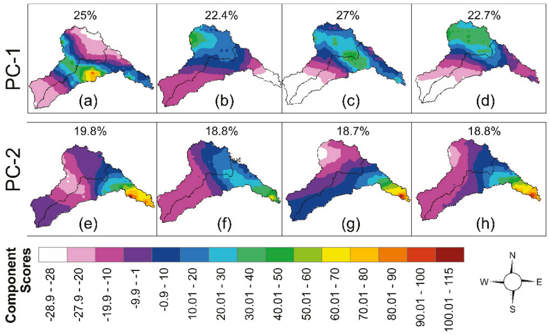


Figure 5. The spatial patterns of the first two PCs for the daily seasonal precipitation across the study area. The explained variance % is presented on each panel. The component scores are presented by the color distributions across the study area (see color bar for reference): (**a**) warm-wet (PC-1), (**b**) cold-dry (PC-1), (**c**) cold-wet (PC-1), (**d**) warm-dry (PC-1), (**e**) warm-wet (PC-2), (**f**) cold-dry (PC-2), (**g**) cold-wet (PC-2) and (**h**) warm-dry (PC-2) seasons.

These two PCs cumulatively produced variance of 44.8% in the warm-wet season, 41.2% in the cold-dry season, 45.7% in the cold-wet season and 41.5% in the warm-dry season. For the component scores, the first PC for the warm-wet season, as shown in Figure 5a (with 25% of the explained variance), has high negative scores in the north and southwest of the study region and medium-high positive signals in the central region. Stronger positive and negative signals exhibit an affinity of higher variation in the precipitation amount, while weaker scores indicate low variability. The second PC for the warm-wet season, as shown in Figure 5e (with 19.8% of the explained variance), has mostly positive component scores in the southeast, except for medium and strong negative signals in several southeast and central regions, respectively. In the cold-dry season, we observe higher negative variability in the southwest based on PC1, as shown in Figure 5b and in small areas of the southeast. Strong positive signals were detected in the northwest area of the study region. For PC2, high positive component scores were observed in the southeast, as shown in Figure 5f (with 18.8% of the explained variance), whereas strong negative scores occurred in the southwest area. In the cold-wet season, the variability in the precipitation amount was high in the southwest region for PC1 (with 27% of the explained variance in the data), as shown in Figure 5c and high positive signals in the southeast and central areas of the region. For PC2, cold-wet season (with 18.7% of the explained variance), shown in Figure 5g, had strong positive scores in the southeast area of the region but medium to high negative scores in the rest of the study area. In the warm-dry season, high variability was observed in the precipitation in the northern and southwest regions of the area for the PC1 component score (with 22.7% of the explained variance), as shown in Figure 5d. The PC2 had high positive variability in the southwest and medium negative signals in the rest of the region, as shown in Figure 5h. The first two PCs were noted to explain a significant variability for the precipitation data in all seasons.

4.1.2. Agglomerative Hierarchical Clustering (AHC)

We then identified all the potentially homogeneous regions inside the study area. The extent of the homogeneity in these homogeneous regions was identified using different cluster validity indices. The first 20 PCs were employed in the AHC analysis. These PCs were obtained in the previous step using the Python scikit module. The validity of different site/station clusters was evaluated by performing the silhouette score, CH and S Dbw tests. Figure 6 summarizes the cluster validation results. The machine learning algorithms for the cluster validity were run repetitively using different input values for the maximum Euclidean distance, followed by a comparison of the resulting clusters to obtain their validity. According to the results, the optimized partitioning of the data based on the CH Test was investigated while the lowest scores were estimated corresponding to clusters 13, 15, 14 and 14 for the warm-wet, cold-dry, cold-wet and warm-dry seasons, respectively, as shown in Figure 6a. The partitioning based on the S Dbw test suggests that optimized clustering can be performed when the data is partitioned into clusters 13, 13, 14 and 13 for the warm-wet, cold-dry, cold-wet and warm-dry seasons, respectively, as shown in Figure 6b. These clusters were selected based on the lowest S Dbw scores. The partitioning of data into clusters 11, 15, 13 and 13 corresponds to the highest silhouette test scores for the warm-wet, cold-dry, cold-wet and warm-dry seasons, respectively, as shown in Figure 6c.

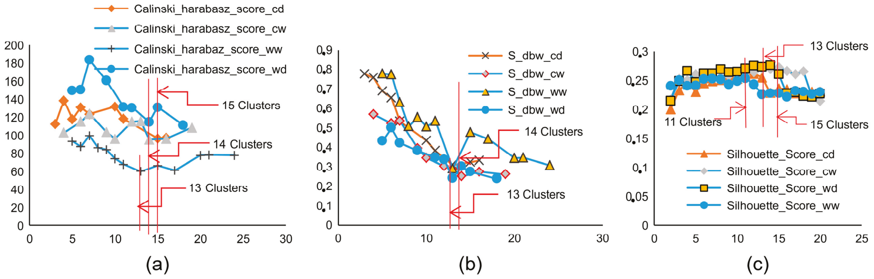


Figure 6. The cluster validity indices/scores for cold-dry (cd), cold-wet (cw), warm-wet (ww) and warm dry (wd) seasons (**a**) Calinski–Harabasz score, the number of clusters valid are marked corresponding to the minimum score in each season (**b**) S Dbw score, the number of clusters valid is marked corresponding to minimum score in each season and (**c**) Silhouette Score the number of clusters valid is marked corresponding to maximum score in each season.

Based on Figure 6, all validity measures agree with the same number of clusters. Therefore, based on the recommendations of previous studies [70], we partitioned the stations for optimized clustering as clusters 13, 13, 14 and 13 for the warm-wet, cold-dry, cold-wet and warm-dry seasons, respectively. The members/stations in different clusters were obtained via the truncation of the dendrogram at the Euclidean distances that correspond to the optimized clusters. The dendrograms were obtained from the AHC of the selected PCs. The maximum Euclidean distances were identified as corresponding to the proposed cluster numbers, where Figure 7 shows the cut-off/truncation bars for each season. Based on this truncation, we identified the groups of stations/sites in each cluster. Figure 7 shows the dendrograms that present the cluster trees for all four seasons.

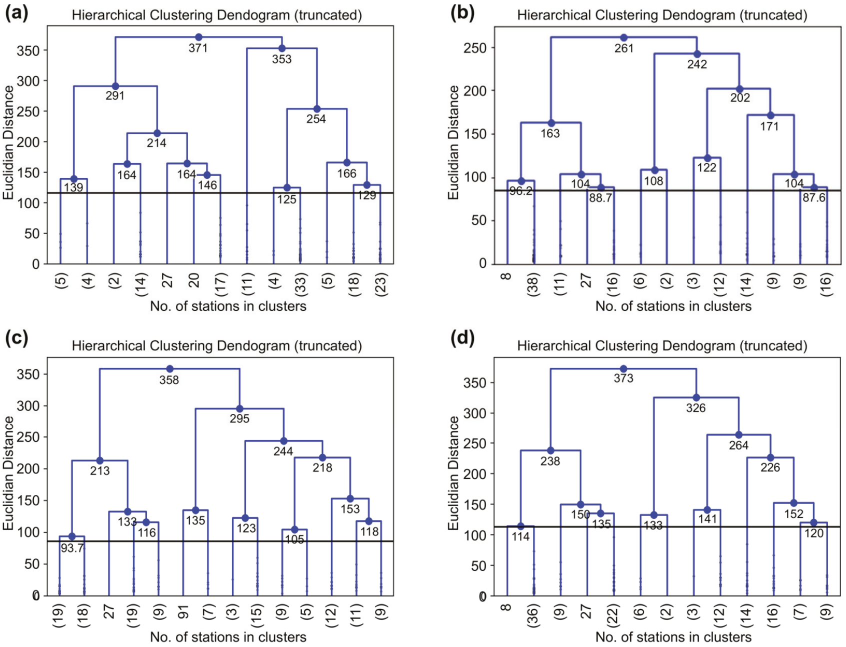


Figure 7. Dendrograms with cut-off bars to group the stations in clusters as a function of the Euclidean distance, the numbers in bracket indicate the number of branches (group of station) in a truncated branch of Dendogram: (**a**) warm-wet, (**b**) cold-dry, (**c**) cold-wet and (**d**) arm-dry seasons. The truncation is done through black cut-off Bar corresponding to the Euclidian distance for the required optimum number of clusters.

4.1.3. Climate Zones and Reference site

This study reveals useful details on the efficiency of machine learning algorithms when formulating large, homogeneous regions of precipitation for different seasons. Homogeneous precipitation sites were identified based on the validated number of cluster-values. These station clusters were then plotted on a study area map, as shown in Figure 6. All homogeneous regions are differentiated by different colors to show the general spatial patterns associated with precipitation, the approximate boundaries are drawn on the maps to allow for a clearer representation. The clusters of stations transformation into different climate zones for all seasons have been shown in Figure 8.

The Jhelum and Chenab river basins are partitioned into clusters 13, 13, 14 and 13 for the warm-wet, cold-dry, cold-wet and warm-dry seasons, respectively, based on the analysis. There were three outliers in the clusters of the warm-wet season, one outlier in the clusters of the cold-dry season, one outlier in the cold-wet and one outlier in the warm-dry season. After merging the outliers with the neighboring clusters, the final demarcations of the basins were mapped for every season. The basins are finally classified into 10, 12, 13 and 13 climate zones for warm-wet, cold-dry, cold-wet and warm-dry seasons. The reference station was identified in each climate zone for the selection of the best-correlated GCM, using the past performance analysis of the GCM at the reference station in each climate zone.

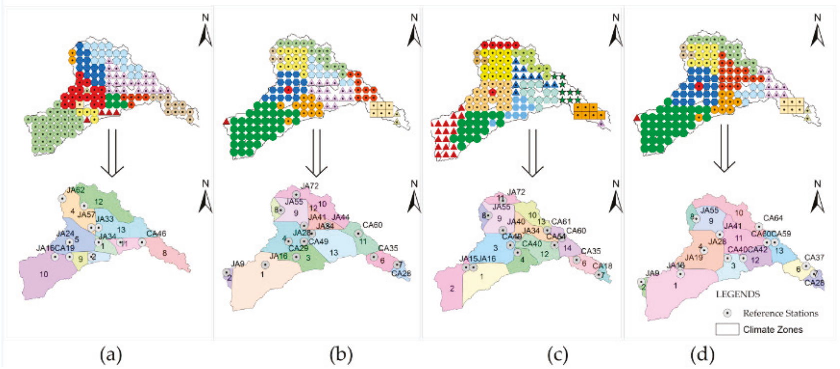


Figure 8. Climate zones in the Jhelum and Chenab river basins in the (**a**) warm-wet, (**b**) cold-dry, (**c**) cold-wet and (**d**) warm-dry seasons.

4.2. GCM Selection

We investigated the capabilities of 21 CMIP5 GCMs to reconstruct the highly resolved observation data for precipitation and, using Pearson correlation analysis, identified the most correlated GCMs for a dependable climate projection in a climate zone. The GCMs were selected in every climate zone of the study area using the GCM and APHRODITE seasonal precipitation data at the reference station. The reference stations in every climate zone for every season have been shown in Figure 8. These selected GCMs were then used for the projections of changes in the precipitation trends at every grid point throughout the respective zone in the river basins. Figure 9 shows the spatial distribution of the highly correlated GCMs in each climate zone. For the warm-wet season, we selected the BNU-ESM, MIROC5, CESM, IPSL-CM5A-MR and GFDL-ESM2G GCMs. For the cold-dry season, we selected the CCSM4, MIROC-ESM, IPSL-CM5A-LR, MIROC5 and MPI-ESM-LR GCMs. For the cold-wet season, we selected the MRI-CGCM3, MIROC-ESM, NorESMI, CESM, IPSL-CM5A-MR and GFDL-ESM2G GCMs. For the warm-dry season, we selected the IPSL-CM5A-MR, BNU-ESM, GFDL-CM3, bcc-CSM1-1, inmcm4 and GFDL-CM3 GCMs. The description of the aforementioned GCMs can be checked from the Table S1 in supplementary documents.

4.3. Validation of Selected GCMs

The selected GCMs and simple mean of MME (21 GCMs) daily precipitation data at every grid station were resampled for the seasonal cycles. The KS test was applied to every grid station and the data distribution was compared with the APHRODITE data distribution. p-values < 0.05 demarcate between the null and alternative hypotheses. When the p-value < 0.05, the similar distribution hypothesis is rejected and vice versa. Figure 10 shows the spatial distribution of the p-value for the KS test. The result shows that the selected GCMs present a higher degree of similarity in the daily seasonal distribution to the APHRODITE distribution as compared with the simple mean of MME distribution. The results show that 45% of the time-series distribution of the selected GCMs' datasets did not fall in the rejection zone (when $\alpha = 5\%$), on the other hand, when using the datasets of simple mean MME, 28% of the datasets did not fall in the rejection zone for the warm-wet season. 76% of the time-series distribution of the datasets of the selected GCMs did not fall in the rejection zone (when $\alpha = 5\%$) for the cold-dry season, whereas only 55% of the datasets were not in the rejection zone when using the datasets of simple mean MME. For the cold-wet season, 76% of the time-series distribution of the selected GCMs' datasets did not fall in the rejection zone (when $\alpha = 5\%$), whereas 0% of the datasets were not in the rejection zone when the simple mean MME was tested. For the warm-dry season, 73% of the time-series distribution of the selected GCMs' datasets did not fall in the rejection zone (when $\alpha = 5\%$), whereas only traces of 10% of the datasets were not in the rejection zone when the simple mean MME was tested. The results validated the GCM selection procedure when assessed against past performance.

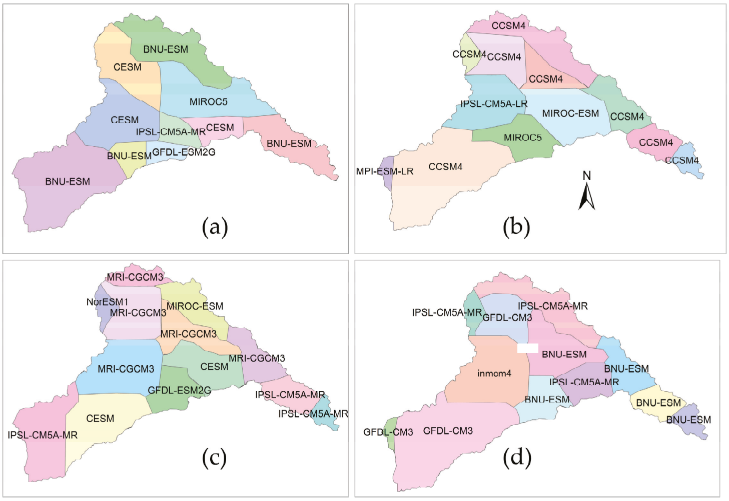


Figure 9. The spatial distribution of the highly correlated general circulation models (GCMs) with the observed data in each respective climate zone for the (**a**) warm-wet, (**b**) cold-dry, (**c**) cold-wet and (**d**) warm-dry seasons.

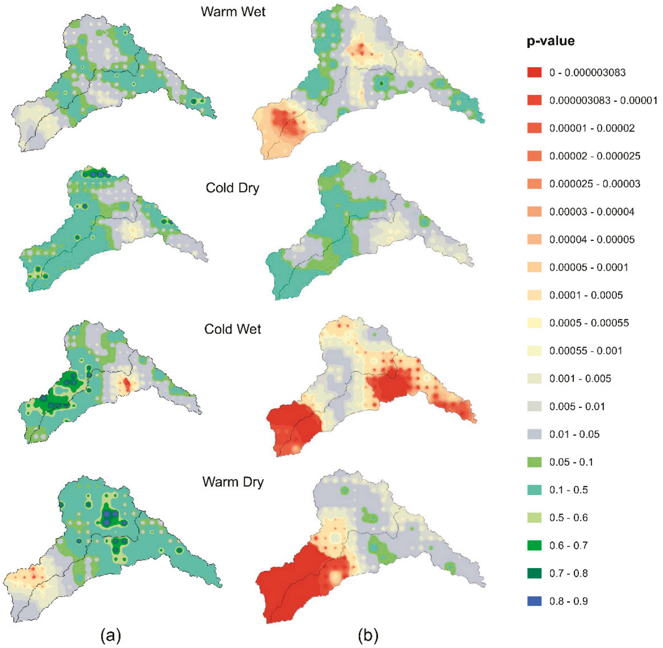


Figure 10. The *p*-values for the K-S Test comparing the APHRODITE data with selected GCMs data and conventional composite mean data of 21 GCMs for warm-wet, cold-dry, cold-wet and warm-dry season (**a**) Using Selected GCMs (**b**) Simple mean Multimodel Ensemble. The green color bands are presenting the zones where the null hypothesis of "similar distribution" is not rejected.

4.4. Seasonal Precipitation Trend Projection

The significance of seasonal precipitation trends for the period of 2006–2099, as detected by the MK trend detection test, for forcing scenarios RCP 4.5 and 8.5, are presented in Figure 11. The distribution of *p*-value spatially for the MK test has been presented. Spatial patterns of the estimated Sen's slope exhibit negative and positive slopes for the trends in different seasons, as shown in Figure 12. The significance of the trends was identified via the MK test of trend detection, as shown in Figure 11. For the warm-wet season, with the RCP 4.5 forcing scenario (refer to Figure 12a), 28% of the total study region yielded very strong and 5% of the total study region yielded strong evidence for decreasing precipitation at values of 2.2–3.29 mm yr^{-1} over the central and northwest of the area. For RCP 4.5, 67% of the region showed no or weak decreasing trends. However, for the RCP 8.5 (Figure 12e), weak evidence for increasing trends was detected in the central and southeast region at 0.5–2 mm yr^{-1}, except for certain areas in the central to western regions, where we detected a non-significant decreasing trend, with a slope of 1–3.29 mm yr^{-1}. Figure 11 depicts the *p*-values for the MK trend detection, which shows significant and non-significant trends.

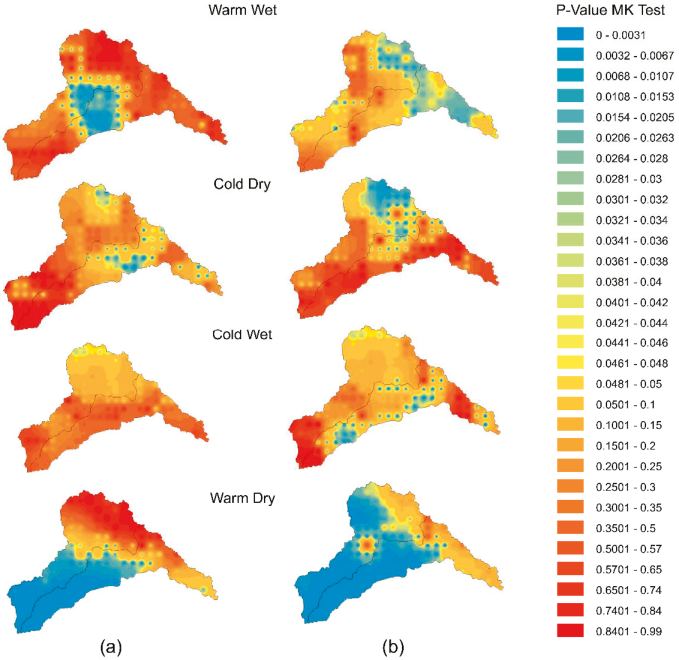


Figure 11. The *p*-values of the MK test for the best-correlated models for the period (2006–2099) for warm-wet, cold-dry, cold-wet and warm-dry for (**a**) Forcing Scenario of RCP 4.5 (**b**) Forcing Scenario of RCP 8.5. The blue and green shades are depicting that the Null Hypothesis of "no trend" is rejected.

For the cold-dry season and RCP 4.5 (Figure 12b), nearly all the regions of the study area had weak, increasing trends with Sen's slopes of 0–0.5 mm yr^{-1}, which are not significant, as depicted in Figure 11. On the other hand, for RCP 8.5 (Figure 12f), weak decreasing trends were detected in the central and southwest regions and certain areas of the southeast, with Sen's slopes of 0.2 mm yr^{-1}. Here, 80% of the total area had weak evidence of increasing trends at 0.5–1.5 mm/season/year.

For the cold-wet season for the RCP 4.5 (Figure 12c), the weak decreasing trend, with Sen's slopes of 0.2–0.5 mm yr^{-1}, was detected in the central and southwest parts of the area. High and weak increasing trends were detected in the northwest and southeast parts of the area, respectively. The positive slope varies between 0.5 and 2 mm yr^{-1}.

For the cold-wet season in the RCP 8.5 (Figure 12g), a non-significant decreasing trend was detected with Sen's slopes varying between 0.5 and 0.2 mm yr^{-1} in the central area, whereas in other regions, a weak, increasing trend was detected with slopes varying between 0.5 and 2 mm yr^{-1}.

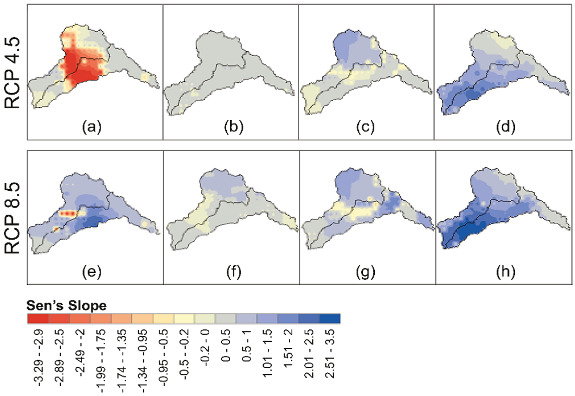


Figure 12. The seasonal precipitation trends in the study area for (**a**) warm-wet RCP 4.5, (**b**) cold-dry RCP 4.5, (**c**) cold-wet RCP 4.5, (**d**) warm-dry RCP 4.5, (**e**) warm-wet RCP 8.5, (**f**) cold-dry RCP 8.5, (**g**) cold-wet RCP 8.5 and (**h**) warm-dry RCP 8.5.

Unlike for the warm-dry season and RCP 4.5 (Figure 12d), there is strong evidence for an increase in the precipitation for the whole study area with *p*-values from the MK trend detection at less than 0.05 for nearly the entire study area, with the Sen's slope indicating an increase in the precipitation rate with values varying between 1.5 and 3.5 mm yr^{-1}. The same is the case for RCP 8.5 in the warm-dry season based on strong evidence of an increase in the precipitation throughout the entire area, with Sen's slope values varying between 1.5 and 3.5 mm yr^{-1} (Figure 12h). Figure 11 shows the *p*-values for the MK test for the best-correlated models (2006–2099) for RCPs 4.5 and 8.5.

5. Conclusions

Previous studies have not currently arrived at a consensus on a universally accepted method and criteria used for GCM selection [14,78]. Studies continue to investigate the methods, whether dynamic or statistical, to reduce the uncertainty in predicting climate change impacts. Intending to reduce the uncertainty, we presented a novel and flexible framework for the selection of GCMs in homogeneous climatic zones, which are based on daily seasonal precipitation data statistics spanning the baseline/historical period (1970–2005) for the Jhelum and Chenab river basin study areas. The GCM selected in each of the climate zones of every season can reproduce the climate distribution patterns in the study area. This has been proved by comparatively analyzing the precipitation data sets of GCMs selected and the simple composite mean of the ensemble of the 21 CMIP5 GCMs. Using these homogeneous precipitation regions, we suggest a different approach for selecting the GCMs. The GCMs were selected based on prior performance and compared with the highly resolved and gridded APHRODITE data. The GCM selection was validated for its performance to emulate the spatial precipitation patterns during the baseline period (1970–2005) for the study region. We sampled the daily precipitation data for the projected period using the selected GCM. The trends in the precipitation variability, based on the MK trend detection statistics, show that the regions in the Jhelum and Chenab river basins have negative, positive and no trends from 2006 to 2099.

The Jhelum and Chenab river basins are scarcely gauged regions; hence, the APHRODITE gridded datasets were used in this study because this type of climate zoning requires a dense network of climate observation stations. Numerous previous studies have verified the reliability of the APHRODITE data in this region [22,40]. However, at high altitudes, the deviation between the gridded data and in situ observations is more than that of flat regions due to a reduced number of gauging stations in mountainous areas, which poses a challenge to precise interpolation [79]. Uncertainties in the forcing datasets persist due to the scarcity of gauging stations in high elevation zones. Furthermore, gauging stations in valleys cannot represent peripheral higher elevation zones due to the high vertical lapse rate of precipitation [41]. Currently, there is no consensus among the climate research community regarding the methods of bias correction for high elevation areas. The comparative analysis shows that the APHRODITE correlates well with the observed dataset of some gauging stations at high altitudes, leading to the assumption of having sufficient accuracy for the regionalization process. Nevertheless, this regionalization framework is equally applicable to other datasets having comparatively higher resolution and accuracy.

All the analyses were conducted using the Python programming language, which is known to be powerful in machine learning. Several machine learning modules from the scikit-learn library, such as Principal Component Analysis (PCA), Agglomerative Hierarchical Clustering (AHC) and clustering validity indices, correlation coefficient and the KS test, were used in the analysis. All these machine learning algorithms were compiled to develop a program for regionalization and climate change trend detection. The program can be provided to the climate research community to augment decision- and policymaking for water resource planning and management.

Potential forecasts from the GCMs, however, are fundamentally uncertain and decision-makers still find it difficult to understand or use such predictions of climate change. The major cause of this uncertainty in the assessments derives from the inherent uncertainties in GCM outputs [80–83]. Based on the results and interpretations presented in this study, we can draw the following conclusions:

1) Due to its highly volatile hydroclimatic conditions, this area of Pakistan poses major scientific challenges; hence, investigations require complex methods. Multivariate techniques, such as PCA and Agglomerative Hierarchical Clustering (AHC) algorithms, are used to develop decisive precipitation statistics and delineate homogeneous precipitation regions, respectively. The entire study area was divided into 13 homogeneous precipitation regions for the warm-wet season, 13 homogeneous precipitation regions for the cold-dry season, 14 homogeneous precipitation regions for the cold-wet season and 13 homogeneous precipitation regions for the warm-dry season. The reference station, which is representative of the respective homogeneous precipitation regions, was obtained in each homogeneous precipitation region for GCM selection. Seasonal rainfall characteristics were incorporated to define the homogeneous climate regions to design the climate zones in the river basins of Pakistan. Representative rainfall station/grid points were selected in each climate zone for the climate change impact assessment. This study provides an objective demarcation structure for homogeneous precipitation regions based on the statistical characteristics of seasonal precipitation.
2) The best-correlated GCM was identified in each climate zone in the Jhelum and Chenab river basins, followed by data extraction. This selection was validated and compared spatially with the KS test. Two schemes of the daily precipitation data were derived: (1) the precipitation data sampled from the selected GCMs based on this framework and (2) the precipitation data from the MME mean (21 models of the CMIP5 mean). The KS test was used to measure the compatibility of the observed data with data from scheme 1 and then the data from scheme 2. The results clearly show the improved performance of the present framework for the selected GCMs compared with the conventional method involving the equally weighted means of all the available GCMs.
3) The precipitation pattern can represent the climatic dynamics because the atmospheric circulation covariates have a strong correlation with the precipitation patterns in the region [2]. The daily projected precipitation data (2006–2099), synthesized for the regions of the Jhelum and Chenab river basins, are publicly available. The creation of homogeneous climatic zones accommodates a deeper understanding of the dynamic spatio-temporal precipitation variability over a given

region. Infrastructure development and climate forecasting are based on effective regionalization using reliable estimates of climate variables. The region should be conditionally considered as climatically homogeneous to proceed with the selection of GCMs for the climate change impact assessments.

4) The projections of the changes in the precipitation trends were presented based on two scenarios, that is, RCP 4.5 and RCP 8.5. For the RCP 4.5, a significant positive precipitation trend was observed for the warm-dry season, while insignificant positive trends were observed for the cold dry and cold wet seasons. A negative precipitation trend was observed throughout the entire study area, in which 40% of the area had a significant negative trend for the warm-wet season. For the RCP 8.5, the warm-dry season again exhibited a significant positive precipitation trend, whereas insignificant positive trends were observed for the warm-wet, cold-dry and cold-wet seasons. The effect of the precipitation change trends may have greater implications on water resources and its management. This unique study provides spatial changes in patterns, as well as temporal changes. Strong increasing and decreasing signals occurred during the warm-dry season for both scenarios in the river basins. Precipitation trends were detected within this framework, which were mapped spatially across the entire study region.

5) Future studies should redefine the homogeneous precipitation region and change detection using the projected data to obtain the hydrological response in a basin using this projected input data for the rainfall. The same framework, with a set of machine learning modules, can be employed to select and detect precipitation trends using the next generation of GCMs. More than 125 years of precipitation data of daily scale, were used for the analyses in this study, which depicts the strength of machine learning.

There is extensive literature available, addressing the uncertainties in climate projections through climate models [5,17]. The uncertainty in predicting the near future day to day weather of dynamic chaotic weather systems has been discussed by meteorologist Edward N Lorenz [16], in which he emphasized the theory of chaos, that is, the importance of the sensitivity of initial state simulation and parametrization of a model for plausible weather forecast. The knowledge of the precise initial state of the weather conditions is essential for the robust whether forecast, furthermore, it is also useful to predict the climate having initial conditions of weather. However, the predictability of the weather is limited to few days due to the reason of atmospheric dynamics. Weather patterns are affected by atmospheric oscillations such as El Nino and La Nina. Climate projections are done using the heat energy changes on Earth's surface and the changes in greenhouse gas emissions in the atmosphere. The projections of these changes to eventually project the climate is significantly easier than the weather forecasting of a week. To summaries, long term changes in atmospheric composition are significantly more predictable than the short term weather forecast [84].

The sources of confidence in future projections by the models are their physical basis and their performance in simulating the past/historical climate. According to IPCC, the models have been proven as important tools to project the future climate [84]. Their performance in representing large scale climate features are extensively evaluated through the comparative analysis of its simulations with the observations and they have been used to predict the ancient climate such as the warm Holocene period, the last glacial maximum and the last ice age [84].

The paper is envisioned to provide a framework that can easily be replicated to project the climate change trends for an area by utilizing the stochastic analysis strengths of machine learning and the available observed and GCMs' data. It tends to be an important reference to those who are working on impact modeling. However, it does not aim to discuss the dynamic processes and associated uncertainties present in all GCMs. These are presented in the literature and their simulation outputs have been used in the analysis. To gain a higher confidence level, the end-users of the present framework are suggested to use large ensemble of GCMs while selecting a suitable model in each zone of the study area. The codes prepared in this study are provided upon request.

Further studies are required to analyze the seasonal climate shift and how accurately the GCM outputs can project the present climatic zones, which are delineated based on the observed gridded historical data.

Supplementary Materials: The following are available online at http://www.mdpi.com/2076-3417/10/19/6878/s1, Table S1: Descriptions of the GCMs used in the study [43].

Author Contributions: Conceptualization, A.N., M.S. and H.F.G.; methodology, A.N., M.S. and S.A.; software, S.H.; validation, S.A., S.H. and H.F.G.; formal analysis, A.N. and M.S.; investigation, A.N.; resources, H.F.G.; data curation, A.N., M.S. and S.A.J.; writing—original draft preparation, A.N.; writing—review and editing, S.H, S.A. and H.F.G.; visualization, A.N. and S.A.J.; supervision, H.F.G.; project administration, S.S.J. and S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We highly acknowledge the support of Jahangir Ali, who downloaded the extracted NEX GDDP data and provided guidance pertaining to the machine learning algorithms.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jamro, S.; Channa, F.N.; Dars, G.H.; Ansari, K.; Krakauer, N.Y. Exploring the Evolution of Drought Characteristics in Balochistan, Pakistan. *Appl. Sci.* **2020**, *10*, 913. [CrossRef]
2. Asong, Z.E.; Khaliq, M.N.; Wheater, H.S. Regionalization of Precipitation Characteristics in the Canadian Prairie Provinces Using Large-scale Atmospheric Covariates and Geophysical Attributes. *Stoch. Environ. Res. Risk Assess.* **2015**, *29*, 875–892. [CrossRef]
3. Christensen, J.; Kanikicharla, K.; Aldrian, E.; An, S.-I.; Fonseca, I.; Castro, M.; Dong, W.; Goswami, P.; Hall, A.; Kanyanga, J.K.; et al. *Climate Phenomena and Their Relevance for Future Regional Climate Change*; IPCC: Geneva, Switzerland, 2013; pp. 1–92.
4. McSweeney, C.F.; Jones, R.G.; Lee, R.W.; Rowell, D.P. Selecting CMIP5 GCMs for downscaling over multiple regions. *Clim. Dyn.* **2015**, *44*, 3237–3260. [CrossRef]
5. Ahmed, K.; Shahid, S.; Chung, E.-S.; Wang, X.; Harun, S.B. Climate Change Uncertainties in Seasonal Drought Severity-Area-Frequency Curves: Case of Arid Region of Pakistan. *J. Hydrol.* **2019**, *570*, 473–485. [CrossRef]
6. Wu, C.; Huang, G.; Yu, H.; Chen, Z.; Ma, J. Impact of Climate Change on Reservoir Flood Control in the Upstream Area of the Beijiang River Basin, South China. *J. Hydrometeor.* **2014**, *15*, 2203–2218. [CrossRef]
7. Ahmed, K.; Sachindra, D.A.; Shahid, S.; Demirel, M.C.; Chung, E.S. Selection of multi-model ensemble of general circulation models for the simulation of precipitation and maximum and minimum temperature based on spatial assessment metrics. *Hydrol. Earth Syst. Sci.* **2019**, *23*, 4803–4824. [CrossRef]
8. Ismail, H.; Kamal, M.R.; Abdullah, A.F.B.; Jada, D.T.; Sai Hin, L. Modeling Future Streamflow for Adaptive Water Allocation under Climate Change for the Tanjung Karang Rice Irrigation Scheme Malaysia. *Appl. Sci.* **2020**, *10*, 4885. [CrossRef]
9. Uamusse, M.M.; Tussupova, K.; Persson, K.M. Climate Change Effects on Hydropower in Mozambique. *Appl. Sci.* **2020**, *10*, 4842. [CrossRef]
10. Touseef, M.; Chen, L.; Masud, T.; Khan, A.; Yang, K.; Shahzad, A.; Wajid Ijaz, M.; Wang, Y. Assessment of the Future Climate Change Projections on Streamflow Hydrology and Water Availability over Upper Xijiang River Basin, China. *Appl. Sci.* **2020**, *10*, 3671. [CrossRef]
11. Yu, Z.; Man, X.; Duan, L.; Cai, T. Assessments of Impacts of Climate and Forest Change on Water Resources Using SWAT Model in a Subboreal Watershed in Northern Da Hinggan Mountains. *Water* **2020**, *12*, 1565. [CrossRef]
12. Dikshit, A.; Pradhan, B.; Alamri, A.M. Short-Term Spatio-Temporal Drought Forecasting Using Random Forests Model at New South Wales, Australia. *Appl. Sci.* **2020**, *10*, 4254. [CrossRef]
13. Zhao, Q.; Ma, X.; Liang, L.; Yao, W. Spatial–Temporal Variation Characteristics of Multiple Meteorological Variables and Vegetation over the Loess Plateau Region. *Appl. Sci.* **2020**, *10*, 1000. [CrossRef]
14. McMahon, T.A.; Peel, M.C.; Karoly, D.J. Assessment of Precipitation and Temperature Data from CMIP3 Global Climate Models for Hydrologic Simulation. *Hydrol. Earth Syst. Sci.* **2015**, *19*, 361–377. [CrossRef]

15. Maxino, C.C.; McAvaney, B.J.; Pitman, A.J.; Perkins, S.E. Ranking the AR4 Climate Models over the Murray-Darling Basin Using Simulated Maximum Temperature, Minimum Temperature and Precipitation. *Int. J. Climatol.* **2008**, *28*, 1097–1112. [CrossRef]
16. Lorenz, E.N. Predictability: A Problem Partly Solved. In Proceedings of the Seminar on Predictability, Shinfield Park, Reading, UK, 4–8 September 1995.
17. Hawkins, E.; Sutton, R. The Potential to Narrow Uncertainty in Projections of Regional Precipitation Change. *Clim. Dyn.* **2011**, *37*, 407–418. [CrossRef]
18. Baker, N.; Taylor, P. A Framework for Evaluating Climate Model Performance Metrics. *J. Clim.* **2016**, *29*, 1773–1782. [CrossRef]
19. Chhin, R.; Yoden, S. Ranking CMIP5 GCMs for Model Ensemble Selection on Regional Scale: Case Study of the Indochina Region. *J. Geophys. Res. Atmos.* **2018**, *123*, 8949–8974. [CrossRef]
20. Wilcke, R.A.I.; Bärring, L. Selecting Regional Climate Scenarios for Impact Modelling Studies. *Environ. Model. Softw.* **2016**, *78*, 191–201. [CrossRef]
21. Salman, S.A.; Shahid, S.; Ismail, T.; Ahmed, K.; Wang, X.-J. Selection of Climate Models for Projection of Spatiotemporal Changes in Temperature of Iraq with Uncertainties. *Atmos. Res.* **2018**, *213*, 509–522. [CrossRef]
22. Lutz, A.F.; ter Maat, H.W.; Biemans, H.; Shrestha, A.B.; Wester, P.; Immerzeel, W.W. Selecting Representative Climate Models for Climate Change Impact Studies: An Advanced Envelope-Based Selection Approach. *Int. J. Climatol.* **2016**, *36*, 3988–4005. [CrossRef]
23. Cannon, A.J. Selecting GCM Scenarios that Span the Range of Changes in a Multimodel Ensemble: Application to CMIP5 Climate Extremes Indices. *J. Clim.* **2015**, *28*, 1260–1267. [CrossRef]
24. Azmat, M.; Qamar, M.U.; Huggel, C.; Hussain, E. Future Climate and Cryosphere Impacts on the Hydrology of a Scarcely Gauged Catchment on the Jhelum River Basin, Northern Pakistan. *Sci. Total Environ.* **2018**, 961–976. [CrossRef] [PubMed]
25. Mahmood, R.; Jia, S.; Tripathi, N.K.; Shrestha, S. Precipitation Extended Linear Scaling Method for Correcting GCM Precipitation and Its Evaluation and Implication in the Transboundary Jhelum River Basin. *Atmosphere* **2018**, *9*, 160. [CrossRef]
26. Kim, J.-B.; So, J.-M.; Bae, D.-H. Global Warming Impacts on Severe Drought Characteristics in Asia Monsoon Region. *Water* **2020**, *12*, 1360. [CrossRef]
27. Gu, H.; Yu, Z.; Wang, J.; Wang, G.; Yang, T.; Ju, Q.; Yang, C.; Xu, F.; Fan, C. Assessing CMIP5 General Circulation Model Simulations of Precipitation and Temperature over China. *Int. J. Climatol.* **2015**, *35*, 2431–2440. [CrossRef]
28. Fu, G.; Charles, S.P.; Kirshner, S. Daily Rainfall Projections from General Circulation Models with a Downscaling Nonhomogeneous Hidden Markov Model (NHMM) for South-Eastern Australia. *Hydrol. Process.* **2013**, *27*, 3663–3673. [CrossRef]
29. Johnson, F.; Sharma, A. Measurement of GCM Skill in Predicting Variables Relevant for Hydroclimatological Assessments. *J. Clim.* **2009**, *22*, 4373–4382. [CrossRef]
30. Schaeffer, M.; Selten, F.M.; Opsteegh, J.D. Shifts of Means Are Not a Proxy for Changes in Extreme Winter Temperatures in Climate Projections. *Clim. Dyn.* **2005**, *25*, 51–63. [CrossRef]
31. Srinivasa Raju, K.; Sonali, P.; Nagesh Kumar, D. Ranking of CMIP5-based global climate models for India using compromise programming. *Theor. Appl. Climatol.* **2017**, *128*, 563–574. [CrossRef]
32. Khan, N.; Shahid, S.; Ahmed, K.; Ismail, T.; Nawaz, N.; Son, M. Performance Assessment of General Circulation Model in Simulating Daily Precipitation and Temperature Using Multiple Gridded Datasets. *Water* **2018**, *10*, 1793. [CrossRef]
33. Knutti, R.; Masson, D.; Gettelman, A. Climate Model Genealogy: Generation CMIP5 and How We Got There. *Geophys. Res. Lett.* **2013**, *40*, 1194–1199. [CrossRef]
34. Yokoi, S.; Takayabu, Y.N.; Nishii, K.; Nakamura, H.; Endo, H.; Ichikawa, H.; Inoue, T.; Kimoto, M.; Kosaka, Y.; Miyasaka, T.; et al. Application of Cluster Analysis to Climate Model Performance Metrics. *J. Appl. Meteorol. Climatol.* **2011**, *50*, 1666–1675. [CrossRef]
35. Min, S.-K.; Hense, A. A Bayesian Approach to Climate Model Evaluation and Multi-Model Averaging with an Application to Global Mean Surface Temperatures from IPCC AR4 Coupled Climate Models. *Geophys. Res. Lett.* **2006**, *33*, Ar.4. [CrossRef]

36. Perkins, S.E.; Pitman, A.J.; Holbrook, N.J.; McAneney, J. Evaluation of the AR4 Climate Models' Simulated Daily Maximum Temperature, Minimum Temperature, and Precipitation over Australia Using Probability Density Functions. *J. Clim.* **2007**, *20*, 4356–4376. [CrossRef]
37. Jiang, X.; Waliser, D.E.; Xavier, P.K.; Petch, J.; Klingaman, N.P.; Woolnough, S.J.; Guan, B.; Bellon, G.; Crueger, T.; DeMott, C.; et al. Vertical Structure and Physical Processes of the Madden-Julian Oscillation: Exploring Key Model Physics in Climate Simulations. *J. Geophys. Res. Atmos.* **2015**, *120*, 4718–4748. [CrossRef]
38. Immerzeel, W.W.; Wanders, N.; Lutz, A.F.; Shea, J.M.; Bierkens, M.F.P. Reconciling High-Altitude Precipitation in the Upper Indus Basin with Glacier Mass Balances and Runoff. *Hydrol. Earth Syst. Sci.* **2015**, *19*, 4673–4687. [CrossRef]
39. Lutz, A.; Immerzeel, W.; Kraaijienbrink, P.D.A. *Gridded Meteorological Datasets and Hydrological Modelling in the Upper Indus Basin*; Future Water: Wageningen, The Netherlands, 2014; p. 83.
40. Yatagai, A.; Kamiguchi, K.; Arakawa, O.; Hamada, A.; Yasutomi, N.; Kitoh, A. APHRODITE: Constructing a Long-Term Daily Gridded Precipitation Dataset for Asia Based on a Dense Network of Rain Gauges. *Bull. Am. Meteorol. Soc.* **2012**, *93*, 1401–1415. [CrossRef]
41. Lutz, A.; Immerzeel, W. *Water Availability Analysis for the Upper Indus, Ganges and Brahmaputra River Basins*; Future Water: Wageningen, The Netherlands, 2013; p. 85.
42. ECMWF. European Reanalysis Dataset (ERA5). 2020. Available online: http://climate.copernicus.eu/climate-reanalysis (accessed on 9 September 2020).
43. Global Meteorological Forcing Dataset for Land Surface Modeling. *Research Data Archive at the National Center for Atmospheric Research*; Computational and Information Systems Laboratory: Boulder, CO, USA, 2006. [CrossRef]
44. Taylor, E.K.; Ronald, S.; Meehl, G. An Overview of CMIP5 and the Experiment Design. *Bull. Am. Meteorol. Soc.* **2011**, *93*, 485–498. [CrossRef]
45. Thrasher, B.; Maurer, E.; McKellar, C.; Duffy, P.B. Technical Note: Bias Correcting Climate Model Simulated Daily Temperature Extremes with Quantile Mapping. *Hydrol. Earth Syst. Sci.* **2012**, *16*, 3309–3314. [CrossRef]
46. Sheffield, J.; Goteti, G.; Wood, E.F. Development of a 50-Year High-Resolution Global Dataset of Meteorological Forcings for Land Surface Modeling. *J. Clim.* **2006**, *19*, 3088–3111. [CrossRef]
47. Chen, H.-P.; Sun, J.-Q.; Li, H.-X. Future changes in precipitation extremes over China using the NEX-GDDP high-resolution daily downscaled data-set. *Atmos. Ocean. Sci. Lett.* **2017**, *10*, 403–410. [CrossRef]
48. Carter, T.R. *General Guidelines on the Use of Scenario Data for Climate Impact and Adaptation Assessment*, 2nd ed.; Task Group on Data and Scenario Support for Impact and Climate Assessment (TGICA); Intergovernmental Panel on Climate Change (IPCC): Helsinki, Finland, 2007; p. 66.
49. Gabriele, S.; Chiaravalloti, F. Searching Regional Rainfall Homogeneity Using Atmospheric Fields. *Adv. Water Resour.* **2013**, *53*, 163–174. [CrossRef]
50. Irwin, S.; Srivastav, R.K.; Simonovic, S.P.; Burn, D.H. Delineation of Precipitation Regions Using Location and Atmospheric Variables in Two Canadian Climate Regions: The Role of Attribute Selection. *Hydrol. Sci. J.* **2017**, *62*, 191–204. [CrossRef]
51. Nam, W.; Shin, H.; Jung, Y.; Joo, K.; Heo, J.-H. Delineation of the Climatic Rainfall Regions of South Korea Based on a Multivariate Analysis and Regional Rainfall Frequency Analyses. *Int. J. Climatol.* **2015**, *35*, 777–793. [CrossRef]
52. Rasheed, A.; Egodawatta, P.; Goonetilleke, A.; McGree, J. A Novel Approach for Delineation of Homogeneous Rainfall Regions for Water Sensitive Urban Design—A Case Study in Southeast Queensland. *Water* **2019**, *11*, 570. [CrossRef]
53. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
54. Hosking, J.R.M.; Wallis, J.R. Frontmatter. In *Regional Frequency Analysis: An Approach Based on L-Moments*; Hosking, J.R.M., Wallis, J.R., Eds.; Cambridge University Press: Cambridge, UK, 1997; pp. i–vi.
55. Liu, M.; Huang, Y.; Li, Z.; Tong, B.; Liu, Z.; Sun, M.; Jiang, F.; Zhang, H. The Applicability of LSTM-KNN Model for Real-Time Flood Forecasting in Different Climate Zones in China. *Water* **2020**, *12*, 440. [CrossRef]
56. Zeraatpisheh, M.; Bakhshandeh, E.; Emadi, M.; Li, T.; Xu, M. Integration of PCA and Fuzzy Clustering for Delineation of Soil Management Zones and Cost-Efficiency Analysis in a Citrus Plantation. *Sustainability* **2020**, *12*, 5809. [CrossRef]

57. Chen, Y.; Zheng, B.; Hu, Y. Mapping Local Climate Zones Using ArcGIS-Based Method and Exploring Land Surface Temperature Characteristics in Chenzhou, China. *Sustainability* **2020**, *12*, 2974. [CrossRef]
58. Liu, Q.; Huang, C.; Li, H. Quality Assessment by Region and Land Cover of Sharpening Approaches Applied to GF-2 Imagery. *Appl. Sci.* **2020**, *10*, 3673. [CrossRef]
59. Hsu, K.-C.; Li, S.-T. Clustering Spatial–Temporal Precipitation Data Using Wavelet Transform and Self-Organizing Map Neural Network. *Adv. Water Resour.* **2010**, *33*, 190–200. [CrossRef]
60. Benestad, R.E.; Chen, D.; Mezghani, A.; Fan, L.; Parding, K. On Using Principal Components to Represent Stations in Empirical–Statistical Downscaling. *Tellus A* **2015**, *67*, 28326. [CrossRef]
61. Mendlik, T.; Gobiet, A. Selecting Climate Simulations for Impact Studies Based on Multivariate Patterns of Climate Change. *Clim. Chang.* **2016**, *135*, 381–393. [CrossRef]
62. Carvalho, M.J.; Melo-Gonçalves, P.; Teixeira, J.C.; Rocha, A. Regionalization of Europe Based on a K-Means Cluster Analysis of the Climate Change of Temperatures and Precipitation. *Phys. Chem. Earth Parts A B C* **2016**, *94*, 22–28. [CrossRef]
63. Cowpertwait, P.S.P. A Regionalization Method Based on a Cluster Probability Model. *Water Resour. Res.* **2011**, *47*. [CrossRef]
64. Huth, R.; Beck, C.; Philipp, A.; Demuzere, M.; Ustrnul, Z.; Cahynová, M.; Kyselý, J.; Tveito, O.E. Classifications of Atmospheric Circulation Patterns: Recent Advances and Applications. *Ann. N. Y. Acad. Sci.* **2008**, *1146*, 105–152. [CrossRef] [PubMed]
65. Mimmack, G.M.; Mason, S.J.; Galpin, J.S. Choice of Distance Matrices in Cluster Analysis: Defining Regions. *J. Clim.* **2001**, *14*, 2790–2797. [CrossRef]
66. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
67. Halkidi, M.; Batistakis, Y.; Vazirgiannis, M. On Clustering Validation Techniques. *J. Intell. Inf. Syst.* **2001**, *17*, 107–145. [CrossRef]
68. Caliński, T.; Harabasz, J. A Dendrite Method for Cluster Analysis. *Commun. Stat.* **1974**, *3*, 1–27. [CrossRef]
69. Arbelaitz, O.; Gurrutxaga, I.; Muguerza, J.; Pérez, J.M.; Perona, I. An Extensive Comparative Study of Cluster Validity Indices. *Pattern Recognit.* **2013**, *46*, 243–256. [CrossRef]
70. Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. Understanding of Internal Clustering Validation Measures. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 911–916.
71. Aghakhani, A.A.; Hasanzadeh, Y.; Besalatpour, A.A.; Pourreza-Bilondi, M. Climate Change Forecasting in a Mountainous Data Scarce Watershed Using CMIP5 Models under Representative Concentration Pathways. *Theor. Appl. Climatol.* **2017**, *129*, 683–699. [CrossRef]
72. Xuan, W.; Ma, C.; Kang, L.; Gu, H.; Pan, S.; Xu, Y.-P. Evaluating Historical Simulations of CMIP5 GCMs for Key Climatic Variables in Zhejiang Province, China. *Theor. Appl. Climatol.* **2017**, *128*, 207–222. [CrossRef]
73. Latif, M.; Hannachi, A.; Syed, F.S. Analysis of Rainfall Trends over Indo-Pakistan Summer Monsoon and Related Dynamics Based on CMIP5 Climate Model Simulations. *Int. J. Climatol.* **2018**, *38*, e577–e595. [CrossRef]
74. Wang, W.; Chen, X.; Shi, P.; van Gelder, P.H.A.J.M. Detecting Changes in Extreme Precipitation and Extreme Streamflow in the Dongjiang River Basin in Southern China. *Hydrol. Earth Syst. Sci.* **2008**, *12*, 207–221. [CrossRef]
75. Van Vuuren, D.P.; Edmonds, J.; Kainuma, M.; Riahi, K.; Thomson, A.; Hibbard, K.; Hurtt, G.C.; Kram, T.; Krey, V.; Lamarque, J.-F.; et al. The Representative Concentration Pathways: An Overview. *Clim. Chang.* **2011**, *109*, 5–31. [CrossRef]
76. Mann, H.B. Nonparametric Tests against Trend. *Econometrica* **1945**, *13*, 245–259. [CrossRef]
77. Sen, P.K. Estimates of the Regression Coefficient Based on Kendall's Tau. *J. Am. Stat. Assoc.* **1968**, *63*, 1379–1389. [CrossRef]
78. Smith, I.; Chandler, E. Refining Rainfall Projections for the Murray Darling Basin of South-East Australia—The Effect of Sampling Model Results Based on Performance. *Clim. Chang.* **2010**, *102*, 377–393. [CrossRef]
79. Behnke, R.; Vavrus, S.; Allstadt, A.; Albright, T.; Thogmartin, W.E.; Radeloff, V.C. Evaluation of Downscaled, Gridded Climate Data for the Conterminous United States. *Ecol. Appl.* **2016**, *26*, 1338–1351. [CrossRef]
80. Xu, C.-Y.; Widén, E.; Halldin, S. Modelling Hydrological Consequences of Climate Change—Progress and Challenges. *Adv. Atmos. Sci.* **2005**, *22*, 789–797. [CrossRef]

81. Kay, A.L.; Davies, H.N.; Bell, V.A.; Jones, R.G. Comparison of Uncertainty Sources for Climate Change Impacts: Flood Frequency in England. *Clim. Chang.* **2009**, *92*, 41–63. [CrossRef]
82. Woldemeskel, F.M.; Sharma, A.; Sivakumar, B.; Mehrotra, R. An Error Estimation Method for Precipitation and Temperature Projections for Future Climates. *J. Geophys. Res.* **2012**, *117*. [CrossRef]
83. Zhang, X.; Xu, Y.-P.; Fu, G. Uncertainties in SWAT Extreme Flow Simulation under Climate Change. *J. Hydrol.* **2014**, *515*, 205–222. [CrossRef]
84. Solomon, S.; Qin, D.; Manning, M.; Chen, Z.; Marquis, M.; Averyt, K.B.; Tignor, M.; Miller, H.L. *Climate Change 2007: Working Group I: The Physical Science Basis*; Intergovernmental Panel on Climate Change: Geneva, Switzerland, 2007.

applied sciences

Article

Detection of Precipitation and Fog Using Machine Learning on Backscatter Data from Lidar Ceilometer

Yong-Hyuk Kim [1], **Seung-Hyun Moon** [1] **and Yourim Yoon** [2,*]

[1] School of Software, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea; yhdfly@kw.ac.kr (Y.-H.K.); uramoon@kw.ac.kr (S.-H.M.)

[2] Department of Computer Engineering, Gachon University, 1342 Seongnam-daero, Sujeong-gu, Seongnam-si, Gyeonggi-do 13120, Korea

* Correspondence: yryoon@gachon.ac.kr

Received: 21 August 2020; Accepted: 14 September 2020; Published: 16 September 2020

Abstract: The lidar ceilometer estimates cloud height by analyzing backscatter data. This study examines weather detectability using a lidar ceilometer by making an unprecedented attempt at detecting weather phenomena through the application of machine learning techniques to the backscatter data obtained from a lidar ceilometer. This study investigates the weather phenomena of precipitation and fog, which are expected to greatly affect backscatter data. In this experiment, the backscatter data obtained from the lidar ceilometer, CL51, installed in Boseong, South Korea, were used. For validation, the data from the automatic weather station for precipitation and visibility sensor PWD20 for fog, installed at the same location, were used. The experimental results showed potential for precipitation detection, which yielded an F1 score of 0.34. However, fog detection was found to be very difficult and yielded an F1 score of 0.10.

Keywords: backscatter data; lidar ceilometer; weather detection; machine learning

1. Introduction

The lidar ceilometer is a remote observation device used to measure cloud height at the location in which it is installed. Many studies [1–7] have obtained planetary boundary layer height (PBLH) by analyzing backscatter data: the raw data obtained using a lidar ceilometer. However, there is considerable room for improvement due to the limited accuracy of past methodologies. In the past, backscatter data from a lidar ceilometer were used primarily for PBLH measurements, but recently they have been used for radiation fog alerts [8], optical aerosol characterization [9], aerosol dispersion simulation [10], and studies of the relationship between cloud occurrence and precipitation [11].

Machine learning techniques have been actively applied to the meteorology field in recent years. For example, they are used for forecasting very short-range heavy precipitation [12,13], quality control [14,15] and correction [15–18] of observed weather data, and predicting winter precipitation types [19].

This study attempts to conduct an unprecedented analysis of backscatter data obtained using a lidar ceilometer. Beyond the conventional use of backscatter data for the analysis of PBLH measurement, their correlations with weather phenomena are analyzed and weather detectability is examined. The weather phenomena of precipitation and fog, which are expected to affect backscatter data, were examined. Cloud occurrence is related to precipitation [11], and backscatter measurements can be used to predict fog [8]. Machine-learning techniques are applied in weather detection. To detect the aforementioned weather phenomena (precipitation and fog) from the backscatter data obtained from the lidar ceilometer, three machine learning models: random forest, support vector machine, and artificial neural network were applied.

This paper is organized as follows: Section 2 describes the three machine learning models used in this study. Section 3 introduces backscatter data obtained from the lidar ceilometer, and observational data of precipitation and fog. Section 4 presents machine learning methods for detecting the weather phenomena. Finally, conclusions are drawn in Section 5.

2. Preliminaries

2.1. Random Forest

Random forest, an ensemble learning method used in classification and regression analysis, is designed to output the mode of the classes or the average forecast value of each tree by training multiple decision trees. The first proper random forest was introduced by Breiman [20]. To build a forest of uncorrelated trees, random forest uses a CART (Classification and Regression Tree) procedure combined with randomized node optimization and bagging. The two key elements of random forest are the number of trees and the maximum allowed depth. As the number of random trees increases, random forest generalizes well, but the training time increases. The maximum allowed depth is the number of nodes from the root node to the terminal node. Under-fitting may occur if the maximum allowed depth is small, and over-fitting may occur if it is large. This study set the number of trees to 100 and did not limit the maximum allowed depth.

2.2. Support Vector Machine

Support vector machine (SVM) [21] is a supervised learning model for pattern recognition and data analysis and is mainly used for classification and regression analysis. When a binary classification problem is given, SVM creates a non-probabilistic binary linear classification model for classifying data depending on the category it belongs to. SVM constructs a hyperplane that best separates the training data points with the maximum-margin. In addition to linear classification, SVM can efficiently perform nonlinear classification using a kernel trick that maps data to a high dimensional space.

2.3. Artificial Neural Networks

Artificial neural network (ANN) [22,23] is a statistical learning algorithm inspired by the neural network of biology. ANN generally refers to a model with neurons (nodes) forming a network through the binding of synapses that have a problem-solving ability by changing the binding force of synapses through training. This study used multilayer perceptron (MLP). The basic structure of ANN is composed of the input, hidden, and output layers, and each layer is made up of multiple neurons. Training is divided into two steps: forward and backward calculations. In the forward calculation step, the linear function composed of the weights and thresholds of each layer is used for calculation, and the result is produced through the nonlinear output function. Thus, ANN can perform nonlinear classification because it is a combination of linear and nonlinear functions. In the backward calculation step, it seeks the optimal weight to minimize the error between the predicted and target value (the answer).

3. Weather Data

Three types of weather data were obtained from Korea Meteorological Administration for this study [24]. This section describes the details of each dataset.

3.1. Backscatter Data from Lidar Ceilometer

Backscatter data were collected by a lidar ceilometer installed in Boseong, South Korea, from 1 January 2015 to 31 May 2016. CL51 [25] is a ceilometer manufactured by Vaisala. The CL51 ceilometer can provide backscatter profile and detect clouds up to 13 km, which is twice the range of the previous model CL31. Table 1 gives the basic information on backscatter data. Missing backscatter data were not used. The measurable range of CL51 is 15 km and the vertical resolution is 10 m. Therefore, 15,000 backscatter data are recorded in each observation. However, only the bottom 450 data were used considering that the planetary boundary layer height is generally formed within 3 km. The CL51 provided input to our scheme by calculating the cloud height and volume using the *sky-condition algorithm*. The sky-condition algorithm is used to construct an image of the entire sky based on the ceilometer measurements (raw backscatter data) only from one single point. No more details of this algorithm have been released. The mechanical characteristics of ceilometer CL51 are outlined in Table 2.

Table 1. Information on the collected backscatter data.

Field	Value
Observation period	1 January 2015–31 May 2016
Observation interval	approximately 30 s (irregular)
data count	1,162,077
Height ranges	10−4500 m

Table 2. Specification of lidar ceilometer CL51.

Field	Description or Value
Laser source	Indium Galium Arsenide (InGaAs) Diode Laser
Center wavelength	910 ± 10 nm at 25 °C
Operating mode	Pulsed
Surface diameter	0.5 mm
Cloud detection range	0–13 km
Measurement range	0–15 km
Measurement resolution	10 m
Cloud reporting resolution	5 m
Reporting interval	6–120 s, selectable
Measurement interval	6 s

The BL-View software [26] estimates the PBLH from two types of ceilometer data: levels 2 and 3. In level 2, backscatter data are stored at intervals of 16 s after post-processing of cloud-and-rain filter, moving average, application of threshold values, and removal of abnormal values. In level 3, the PBLHs calculated using the level 2 data are stored. As the raw backscatter data and the level-2 data of BL-View have different measurement intervals, the data of the nearest time slot were matched and used.

For the raw backscatter data, the denoising method [27] was applied. The noise was eliminated through linear interpolation and denoising autoencoder [28]. Considering that the backscatter signals by aerosol particles are mostly similar, the relatively larger backscatter signals were removed through linear interpolation. The backscatter data to which linear interpolation was applied was used as input data of the denoising autoencoder. The moving average of backscatter data was calculated and used as input data to denoise the backscatter data. We used the denoised backscatter data in our experiments. More details about the used backscatter data related to denoising and weather phenomenon are given in Appendix A.

3.2. Data from Automatic Weather Station

This study used the data collected from 1 January 2015 to 31 May 2016 from an automatic weather station (AWS) installed in Boseong, South Korea. AWS is a device that enables automatic weather observation. Observation elements include temperature, accumulated precipitation, precipitation sensing, wind direction, wind speed, relative humidity, and sea-level air pressure. In this study, AWS data with 1 h observation interval was used; the collected information is listed in Table 3. The installation information of the AWS in Boseong is outlined in Table 4.

Table 3. Information on the collected AWS data.

Field	Value
Observation period	1 January 2015–31 May 2016
Observation interval	One hour
#data	17,544

Table 4. Information on the used AWS.

Field		Value
Station number		258
Starting date		8 February 2010
Station name		Boseong
Physical address		Yelang-ri, Deokcheok-myeon, Boseong-gun, Jeollanam-do, South Korea
Latitude		34.7633
Longitude		34.6261
Altitude (m)		2.8
Ground height (m)	Barometer	4.3
	Thermometer	1.7
	Wind gauge	10.0
	Rainfall	0.6

As the observation interval of AWS data is different from that of backscatter data, the data of the nearest time slot based on the backscatter data were matched and used. The used elements included precipitation sensing, accumulated precipitation, relative humidity, and sea-level air pressure.

3.3. Data from Visibility Sensor

Visibility data were collected by PWD20 [29] installed in Boseong, South Korea (see Table 5). PWD20 manufactured by Vaisala is a device that is used to observe the MOR (measurement range) and current weather condition. As its observation range is 10–20,000 m, and vertical resolution is 1 m, it allows a determination of long-range visibility. PWD20 can be fixed to various types of towers because the device is short, compact, and lightweight. The mechanical properties of PWD20 are outlined in Table 6.

Table 5. Information on the collected visibility data.

Field	Value
Observation period	1 January 2015–31 May 2016
Observation interval	1–3 min
#data	1,018,122

Table 6. Information on the used PWD20.

Field	Description or Value
Usage	Visibility for weather observation
Observed element	MOR (measurement range)
Observation method	Forward scatter
Observation range	10–20,000 m
Operation environment	−40–+65 °C

Fog reduces visibility below 1000 m, and it occurs at a relative humidity near 100% [30]. The visibility sensor data was used to determine fog presence (low visibility). The criteria for fog were 1000 m or lower visibility for 20 mins, 90% or higher relative humidity, and no precipitation. The AWS data were used for precipitation sensing and relative humidity. As the ceilometer backscatter data, AWS and visibility sensor data have different observation intervals, the data of the nearest time slot were matched and used based on the ceilometer backscatter data.

4. Weather Detection

In this section, we use the denoised backscatter data, cloud volume, and cloud height as training data, and describe how to detect weather phenomena using three machine learning (ML) models: random forest, SVM, and ANN.

4.1. Data Analysis

The observational data of AWS range from 1 January 2015 to 31 December 2016. The performance of learning algorithms may decrease if they are not provided with enough training data. For precipitation, the presence or absence of precipitation in the AWS hourly data was used. In general, the lower the visibility sensor value was, the higher the probability of fog was. Hence, the visibility sensor data were categorized into 1000 m or below, between 1000 m and 20,000 m, and 20,000 m or higher. Table 7 shows that precipitation data account for 6.38% of all data (1120 cases), and Table 8 shows that visibility sensor data that fall into 1000 m or below account for 1.16% of all data (11,082 cases).

Table 7. Statistics of hourly AWS data related to precipitation.

Field	Missing	Precipitation	Non-Precipitation	Total
#data	558	1120	15,866	17,544
Percentage	3.18%	6.38%	90.44%	100%

Table 8. Statistics of visibility data.

Field	20,000 m	[0 m, 1000 m]	(1000 m, 20,000 m)	Total
#data	84,767	11,802	921,553	1,018,122
Percentage	8.33%	1.16%	90.51%	100%

The precipitation data has a value of 0 or 1, indicating only presence or absence, thus forming a binomial distribution. However, the visibility sensor values range from 0 m to 20,000 m and the data distribution can be represented as a histogram in Figure 1. The MOR in the figure indicates the visibility sensor value, and the values at the bottom of the figure are the mean (μ) and standard distribution (σ) of all visibility sensor values. The line indicates a normal distribution, and the values on the *X* axis are the values obtained by adding or subtracting the standard deviation to or from the mean.

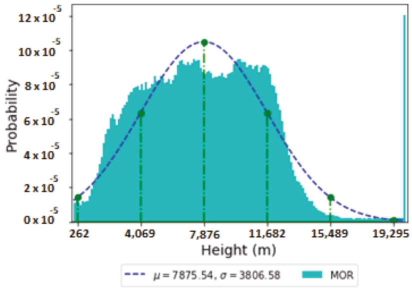


Figure 1. Histogram on visibility data.

The observation interval of ceilometer backscatter data is irregular at approximately 30 s. The observation interval of AWS data is irregular at 1 h, and the observation interval of visibility sensor data is irregular at 1–3 mins. The intervals of observational data were adjusted to those of the backscatter data.

4.2. Training Data Generation

The training data were generated using denoised backscatter data and weather phenomenon (presence/absence) data. For precipitation, the precipitation sensing value of AWS was used. For fog, AWS and visibility data were used to indicate whether fog occurred. As shown in Table 9, the backscatter coefficients of all heights were used for AWS hourly data.

Table 9. Field information on train data for weather phenomenon detection.

Field Information	Description or Value	
Hourly AWS data	1	Month
	2	Day
	3	Backscatter coefficients on heights 10 m, 20 m, . . . , 4500 m
	4	Presence of weather phenomenon (0 or 1)

4.3. Under-Sampling

The number of absent examples was much higher than that of present examples in the training data. Such highly imbalanced data can hinder the training process of machine learning algorithms, making the resulting prediction model rarely predict the present examples. Therefore, we used under-sampling to balance the training data as in [12,31].

The training data comprised data from the first day to the 15th day of each month, and the validation data were composed of data ranging from the 16th to the last day of each month. Random forest was used to find the optimal under-sampling ratio by varying the presence to absence ratio from 1:1 to 1:7. Note that under-sampling is applied only to the training data.

To validate the results, we calculated and compared the accuracy, precision, false alarm rate (FAR), recall (or probability of detection; POD), and F1 score, which are measures that are frequently used in machine learning and meteorology (see Table 10) [32]. Accuracy is the probability that the observed value will coincide with the predicted value among all data ($(a + d)/n$). In precipitation detection, precision is the probability that the predicted precipitation is correct ($a/(a + b)$); the FAR is the number of false alarms over the total number of alarms or predicted precipitation samples ($b/(a + b)$), and recall (or POD) is the fraction of the total amount of precipitation occurrences that were correctly predicted ($a/(a + c)$). In fog detection, precision is the probability that the predicted fog is correct ($a/(a + b)$), the FAR is the number of false alarms over the total number of alarms or predicted fog samples ($b/(a + b)$), and recall (or POD) is the fraction of the total amount of fog occurrences that were correctly predicted ($a/(a + c)$). F1 score is an index that measures the accuracy of validation, and the harmonic mean of precision and recall ($2 \times (precision \times recall)/(precision + recall)$). In imbalanced classification, accuracy can be a misleading metric. Therefore, the F1 score, which considers both precision and recall, is widely used as a major assessment criterion [33,34].

Table 10. Contingency table for prediction of a binary event. The numbers of occurrences in each category are denoted by a, b, c, and d.

Predicted	**Observed (OBS)**		
	Yes	**No**	**Total**
Yes	a	b	$a + b$
No	c	d	$c + d$
Total	$a + c$	$b + d$	$a + b + c + d = n$

In Tables 11 and 12, F1 score is the highest when the under-sampling ratio is 1:2. When the under-sampling ratio is 1:7, accuracy is high, but precision is very low. A high precision is good considering the accuracy in the case of precipitation or visibility sensor phenomenon. However, if precision is high, there is a tendency to only overestimate the corresponding phenomenon. Therefore, we selected the case of the highest F1 score whereby precision and recall were balanced. In other words, we under-sampled the precipitation and fog (low visibility) at the under-sampling ratio of 1:2.

Table 11. Results of precipitation detection according to the under-sampling ratio. The bold number is the best result.

Under-Sampling Ratio	**Accuracy**	**Precision**	**FAR**	**Recall (POD)**	**F1 Score**
1:1	0.8708	0.6468	0.3532	0.2170	0.3250
1:2	0.9025	0.5172	0.4828	0.2507	**0.3377**
1:3	0.9183	0.4327	0.5673	0.2766	0.3374
1:4	0.9277	0.3710	0.6290	0.2979	0.3305
1:5	0.9335	0.3068	0.6932	0.3078	0.3073
1:6	0.9380	0.2534	0.7466	0.3182	0.2821
1:7	0.9420	0.2093	0.7907	0.3346	0.2575
Rand	0.5000	0.0659	0.9341	0.5000	0.1165
W-rand	0.8769	0.0659	0.9341	0.0659	0.0659

Table 12. Results of low visibility detection according to the under-sampling ratio. The bold number is the best result.

Under-Sampling Ratio	**Accuracy**	**Precision**	**FAR**	**Recall (POD)**	**F1 Score**
1:1	0.9472	0.2308	0.7692	0.0593	0.0944
1:2	0.9782	0.0957	0.9043	0.0941	**0.0949**
1:3	0.9844	0.0534	0.9466	0.1291	0.0756
1:4	0.9865	0.0353	0.9647	0.1717	0.0586
1:5	0.9872	0.0283	0.9717	0.2128	0.0500
1:6	0.9876	0.0245	0.9755	0.2620	0.0448
1:7	0.9876	0.0245	0.9755	0.2837	0.0451
Rand	0.5000	0.0126	0.9874	0.5000	0.0246
W-rand	0.9751	0.0126	0.9874	0.0126	0.0126

We also compared our results with two versions of random prediction (see the two bottom rows of Tables 11 and 12): one method called "Rand" evenly predicts the presence or absence of weather phenomenon at random, and the other method called "W-rand" randomly predicts the presence or absence of weather phenomenon with the weights according to the probability of actual observation of weather phenomenon. We could clearly see that random forest with under-sampling is superior to random prediction with respect to F1 score.

4.4. Feature Selection

A large number of input features significantly increases the computation time of machine learning algorithms and requires an enormous amount of training data to ensure sufficient training. There are

452 input features as shown in Table 9, and these need to be reduced. Figures 2 and 3 show the analyses of denoised backscatter data from 1 January 2015 to 31 May 2016 for precipitation and visibility sensor data, respectively. In Figure 2, 'True' indicates the case of precipitation, and 'False' indicates the case of non-precipitation. In Figure 3, 'True' indicates that the visibility sensor value is equal to or lower than 1000 m, and 'False' indicates that the visibility sensor value is higher than 1000 m. The line indicates the backscatter mean value according to height, and the colored part indicates the area of mean ± standard deviation. Above certain heights, it seems difficult to predict the weather phenomena using backscatter data.

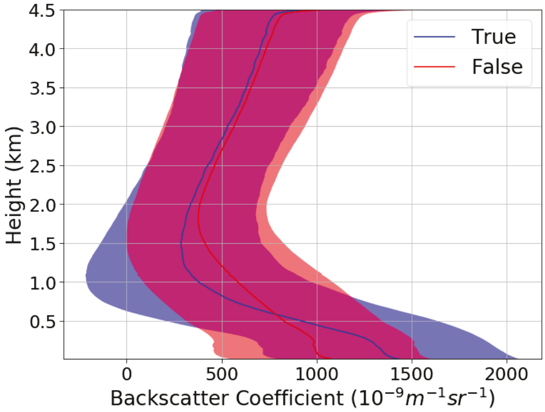


Figure 2. Backscatter data with precipitation (True) and without precipitation (False).

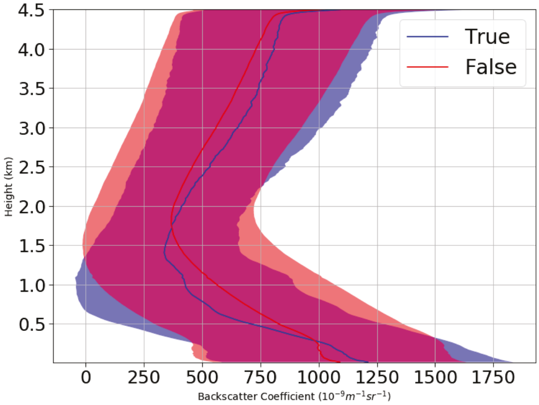


Figure 3. Backscatter data with low visibility (True) or without low visibility (False).

Therefore, we do not have to use all heights to detect weather phenomena. Random forest was applied after categorizing the total height of 4500 m into 10–300 m, 10–600 m, . . . , and 10−4500 m while maintaining the under-sampling ratio at 1:2.

Tables 13 and 14 show that using all the height values ranging from 10 m to 4500 m for precipitation produced satisfactory results. In the case of visibility sensor data, using heights that ranged from 10 m to 3300 m yielded better results than using all the heights that ranged from 10 m to 4500 m. Therefore, the visibility sensor data could yield better results with smaller inputs.

Table 13. Results of precipitation detection according to feature selection. The bold number is the best result.

Height Feature	Accuracy	Precision	FAR	Recall (POD)	F1 Score
Up to 300 m	0.8824	0.2398	0.7602	0.1243	0.1637
Up to 600 m	0.8978	0.3178	0.6822	0.1804	0.2301
Up to 900 m	0.9083	0.3651	0.6349	0.2230	0.2769
Up to 1200 m	0.9106	0.3920	0.6080	0.2385	0.2965
Up to 1500 m	0.9121	0.4144	0.5856	0.2501	0.3120
Up to 1800 m	0.9098	0.4344	0.5656	0.2490	0.3165
Up to 2100 m	0.9108	0.4506	0.5494	0.2564	0.3268
Up to 2400 m	0.9081	0.4591	0.5409	0.2509	0.3245
Up to 2700 m	0.9088	0.4613	0.5387	0.2535	0.3272
Up to 3000 m	0.9050	0.4873	0.5127	0.2498	0.3303
Up to 3300 m	0.9063	0.4852	0.5148	0.2429	0.3325
Up to 3600 m	0.9043	0.4960	0.5040	0.2501	0.3325
Up to 3900 m	0.9032	0.5075	0.4925	0.2503	0.3325
Up to 4200 m	0.9031	0.5131	0.4869	0.2512	0.3373
Up to 4500 m	0.9025	0.5172	0.4828	0.2507	**0.3377**

Table 14. Results of low visibility detection according to feature selection. The bold number is the best result.

Height Feature	Accuracy	Precision	FAR	Recall (POD)	F1 Score
Up to 300 m	0.9655	0.0836	0.9164	0.0405	0.0546
Up to 600 m	0.9771	0.0663	0.9337	0.0630	0.0646
Up to 900 m	0.9798	0.0717	0.9283	0.0854	0.0780
Up to 1200 m	0.9818	0.0812	0.9188	0.1175	0.0960
Up to 1500 m	0.9832	0.0725	0.9275	0.1315	0.0935
Up to 1800 m	0.9830	0.0635	0.9365	0.1150	0.0818
Up to 2100 m	0.9825	0.0729	0.9271	0.1180	0.0901
Up to 2400 m	0.9822	0.0781	0.9219	0.1193	0.0944
Up to 2700 m	0.9819	0.0762	0.9238	0.1138	0.0913
Up to 3000 m	0.9812	0.0846	0.9154	0.1128	0.0967
Up to 3300 m	0.9803	0.0912	0.9088	0.1088	**0.0993**
Up to 3600 m	0.9801	0.0866	0.9134	0.1023	0.0938
Up to 3900 m	0.9797	0.0891	0.9109	0.1014	0.0948
Up to 4200 m	0.9791	0.0949	0.9051	0.1008	0.0978
Up to 4500 m	0.9782	0.0957	0.9043	0.0941	0.0949

In the case of precipitation, the number of input features was not reduced by feature selection, and in the case of visibility sensor, the number of input features was reduced to 332, which is not small enough. To train SVM and ANN, we did not use all the heights of the backscatter data. In the case of precipitation, the height intervals of input features were changed from 10 m to 100 m. In other words, we used the backscatter data at 10 m, 110 m, 210 m, . . . , and 4410 m. For visibility sensor, we used the backscatter data at 10 m, 110 m, . . . , and 3210 m. Therefore, 47 input features were used to predict precipitation and 35 features to predict fog.

With our final model of random forest preprocessed by under-sampling and feature selection, we provide some observation of representative cases for precipitation and fog in Appendix B.

Tables 15 and 16 show the results of SVM and ANN. ANN1 is an MLP with one hidden layer whose number of nodes is half of that of the input layer. ANN2 is an MLP with two hidden layers, and the number of nodes at each hidden layer is half that of its input layer. For both precipitation and fog, random forest best classified the weather phenomena, yielding the highest F1 score.

Table 15. Results of precipitation detection according to other ML techniques.

Method	Accuracy	Precision	FAR	Recall (POD)	F1 Score
SMO	0.8777	0.3649	0.6351	0.1606	0.2230
ANN_1	0.8848	0.2122	0.7878	0.5147	0.3005
ANN_2	0.8644	0.5340	0.4660	0.1849	0.2747

Table 16. Results of low visibility detection according to other ML techniques.

Method	Accuracy	Precision	FAR	Recall (POD)	F1 Score
SMO	0.8528	0.2404	0.7596	0.0203	0.0374
ANN_1	0.8867	0.2820	0.7180	0.0311	0.0560
ANN_2	0.8575	0.2334	0.7666	0.0204	0.0376

5. Concluding Remarks

In this study, we made the first attempt to detect weather phenomena using raw backscatter data obtained from a lidar ceilometer. For weather detection, various machine-learning techniques including under-sampling and feature selection were applied to the backscatter data. The AWS provided observational data for precipitation and the visibility data from PWD20 provided observational data for fog.

Our prediction results were not noticeably good, but if we consider the hardness of weather prediction/detection in the literature (e.g., precision and recall are about 0.5 and 0.3 for heavy rainfall prediction, respectively [13], and they are about 0.2 and 0.5 for lightning forecast, respectively [31]), our prediction results showed potential for precipitation detection (in which precision, recall, and F1 score are about 0.5, 0.2, and 0.3, respectively), but fog detection (in which precision, recall, and F1 score are all about 0.1) was found to be very difficult although it was better than random prediction.

In future work, we expect to improve the accuracy of planetary boundary layer height (PBLH) measurements by classifying backscatter data according to precipitation occurrences.

Author Contributions: Conceptualization, Y.-H.K. and S.-H.M.; methodology, Y.Y. and Y.-H.K.; validation, S.-H.M. and Y.Y.; formal analysis, Y.Y.; investigation, Y.Y. and S.-H.M.; resources, Y.-H.K.; data curation, S.-H.M.; writing—original draft preparation, Y.Y.; writing—review and editing, S.-H.M.; visualization, Y.-H.K.; supervision, Y.-H.K.; project administration, Y.-H.K.; funding acquisition, Y.-H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Technology Development for Supporting Weather Services, through the National Institute of Meteorological Sciences of Korea, in 2017. This research was also a part of the project titled 'Marine Oil Spill Risk Assessment and Development of Response Support System through Big Data Analysis', funded by the Ministry of Oceans and Fisheries, Korea.

Acknowledgments: The authors would like to thank Junghwan Lee and Yong Hee Lee for their valuable helps to greatly improve this paper.

Conflicts of Interest: The authors declare that there is no conflict of interests regarding the publication of this article.

Appendix A Details of the Used Backscatter Coefficients

In this appendix section, we provide some details of the used backscatter data related to denoising and weather phenomenon, through some representative cases. Figure A1 shows an example of raw backscatter data and their denoised ones which were observed at a moment. We can see that noises are successfully removed. Figure A2 shows an extended time-height plot of raw backscatter data and their denoised ones which had been observed for one day (specifically on 18 March 2015 when daily precipitation was 61 mm). Figure A1 can be understood as a cross section at a point of Figure A2. For Figure A2, we chose a day during which both of precipitation and non-precipitation occurred while clouds are presented in both. In the right side of the figure, a gray box means a period that it rains continuously. We could find clear difference between each box boundary point and its adjacent one, but it does not seem easy to distinguish the two phenomena only by the values themselves.

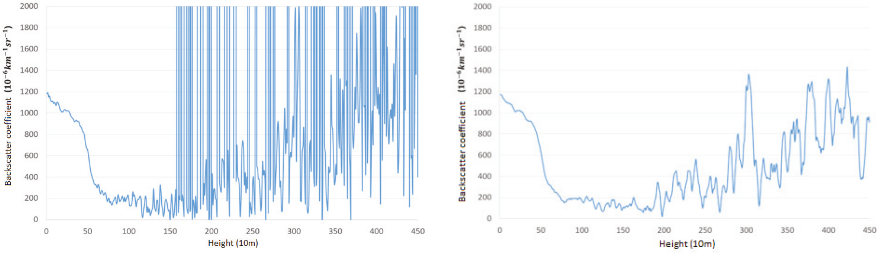


Figure A1. Example data of backscatter coefficients at a moment ((**left**): raw backscatter data and (**right**): denoised backscatter data).

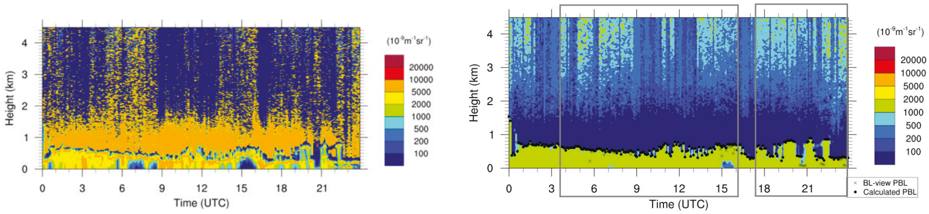


Figure A2. Example time-height plotting of backscatter coefficients on 18 March 2015, when both of precipitation and non-precipitation are mixed: a gray box means a precipitation period ((**left**): raw backscatter data and (**right**): denoised backscatter data).

Figure A3 shows an example of the denoised backscatter data of CL51 and observation range data of PWD20 which had been observed for one day (specifically on 31 March 2015). For the figure, we chose a day during which both of low visibility and not-low visibility occurred while clouds are presented in both. In the left side of the figure, a gray box means a period that visibility is continuously low (i.e., less than 1000 m). Similar to the precipitation case of Figure A2, it is hard to distinguish the two phenomena only by the values themselves. Moreover, we can see that in the middle period it is not easy even to find some difference between the box boundary point and its adjacent one.

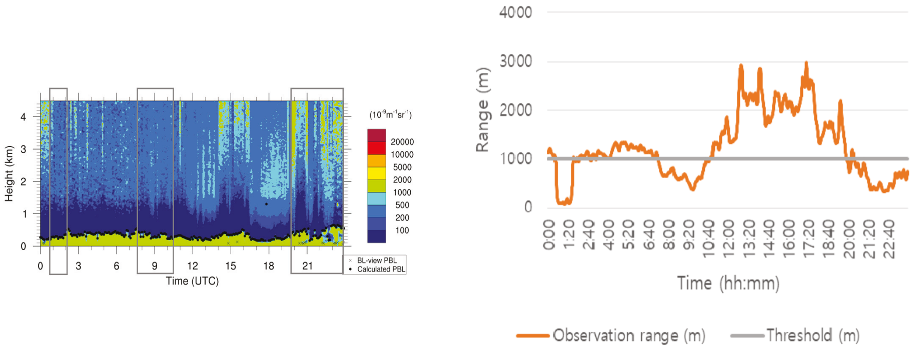


Figure A3. Example time-height plotting of backscatter coefficients on 31 March 2015, when both of low visibility and not-low visibility are mixed: a gray box means a low visibility period ((**left**): denoised backscatter data and (**right**): visibility data).

Appendix B Case Observation

Figure A4 shows an example of detecting precipitation through backscatter data using random forest. In the case of the left side, the observed weather phenomenon at 15:09:36 (hh:mm:ss) on

21 January 2015 is precipitation and the predicted weather phenomenon is also precipitation. In the case of the right side, the observed weather phenomenon at 23:16:48 on 21 January 2015 is non-precipitation, and the predicted weather phenomenon is precipitation. The blue line indicates the mean value of backscatter data according to the observation value, and the colored part is a section where the standard deviation was added or subtracted from the mean. The distributions of backscatter data are generally similar regardless of precipitation. Since the input of our machine learning model is only backscatter data, it seems natural for the model to output the same prediction result for both cases with similar distribution. Hence, this supports the hardness of predicting precipitation.

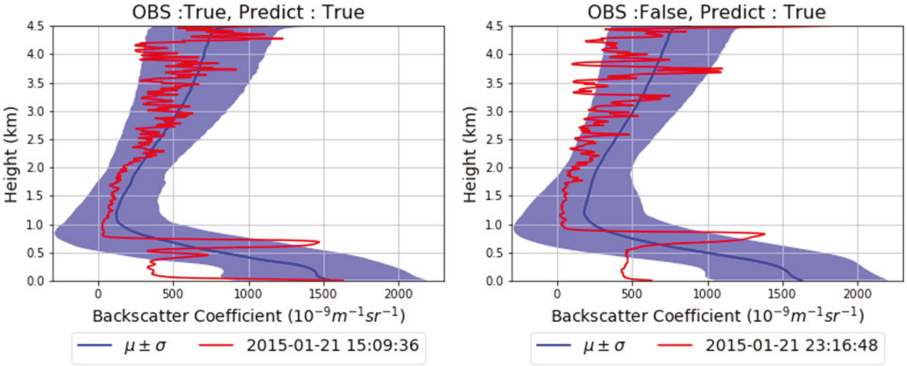


Figure A4. Example of backscatter data predicted as precipitation by machine learning ((**left**): actual precipitation and (**right**): actual non-precipitation).

Figure A5 shows an example of detecting non-precipitation from the backscatter data using random forest. In the case of the left side, the observed weather phenomenon at 23:55:12 on is non-precipitation and the predicted weather phenomenon is also non-prediction. In the case of the right side, the observed weather phenomenon at 15:55:48 on 21 January 2015 is precipitation and the predicted weather phenomenon is non-precipitation. Likewise, the distributions of backscatter data are clearly similar regardless of precipitation. As mentioned above, naturally the machine learning model seems to output the same prediction result for both cases. This also supports that it is difficult to predict non-precipitation.

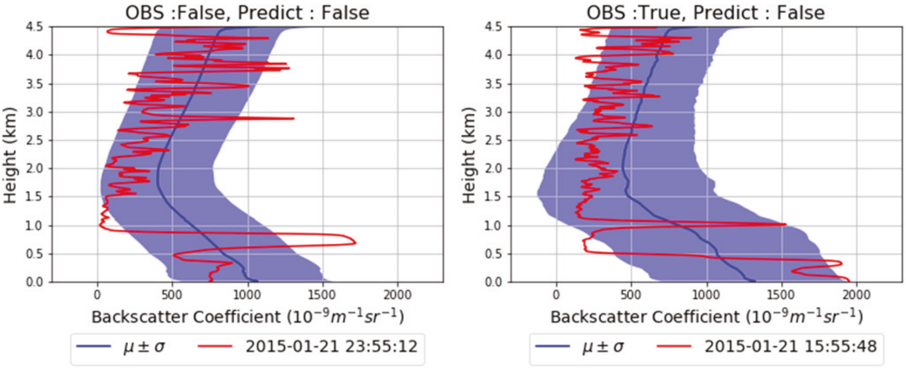


Figure A5. Example of backscatter data predicted as non-precipitation through machine learning ((**left**): actual non-precipitation and (**right**): actual precipitation).

Figure A6 shows an example of detection through the backscatter data using random forest when the visibility sensor value is equal to or less than 1000 m. In the case of the left side, the observed and predicted visibility sensor values at 01:01:46 are both equal to or lower than 1000 m. In the case of the right side, the observed visibility sensor value at 00:46:46 on 22 January 2015 is greater than 1000 m and lower than 20,000 m, and the predicted value is equal to or lower than 1000 m. It is not easy to find clear difference between both cases.

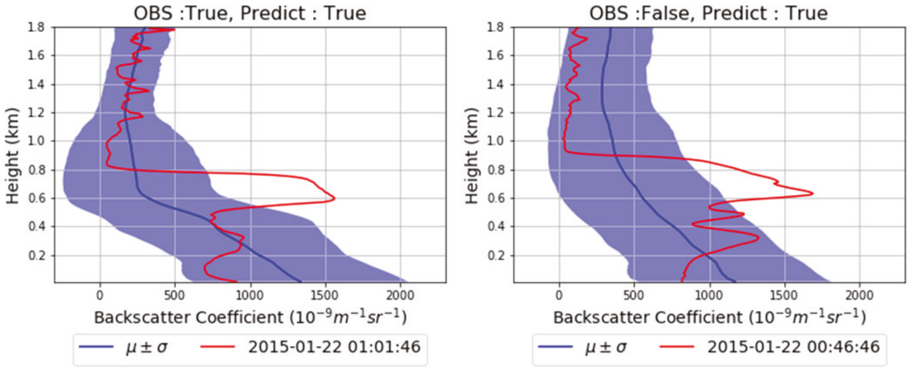


Figure A6. Example of backscatter data predicted as low visibility by machine learning ((**left**): actual low visibility and (**right**): actual not-low visibility).

Figure A7 shows an example of detection from backscatter data using random forest when the visibility sensor value is greater than 1000 m and lower than 20,000 m. In the case of the left side, the observed and predicted visibility sensor values at 00:35:23 on 22 January 2015 are both greater than 1000 m and lower than 20,000 m. In the case of the right side, the observed visibility sensor value at 00:00:00 on 22 January 2015 is 1000 m or lower and the predicted value is greater than 1000 m and lower than 20,000 m. Analogously to the above, we cannot see distinct difference between both cases. It hints that it is very hard to differentiate fog phenomenon by using only backscatter data.

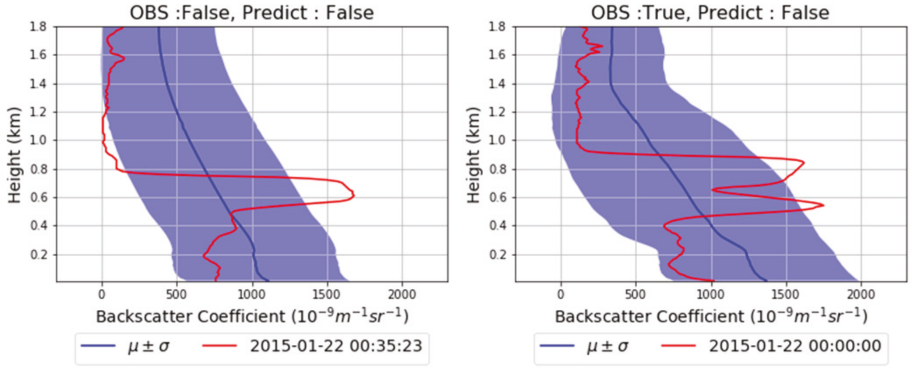


Figure A7. Example of backscatter data predicted as not-low visibility by machine learning ((**left**): actual not-low visibility and (**right**): actual low visibility).

References

1. Münkel, C.; Räsänen, J. New optical concept for commercial lidar ceilometers scanning the boundary layer, Remote Sensing. *Int. Soc. Opt. Photonics* **2004**, *5571*, 364–374.
2. Schäfer, K.; Emeis, S.; Rauch, A.; Münkel, C.; Vogt, S. Determination of mixing layer heights from ceilometer data, Remote Sensing. *Int. Soc. Opt. Photonics* **2004**, *5571*, 248–259.

3. Eresmaa, N.; Karppinen, A.; Joffre, S.M.; Räsänen, J.; Talvitie, H. Mixing height determination by ceilometer. *Atmos. Chem. Phys.* **2006**, *6*, 1485–1493. [CrossRef]
4. de Haij, M.; Wauben, W.; Baltink, H.K. Determination of mixing layer height from ceilometer backscatter profiles, Remote Sensing. *Int. Soc. Opt. Photonics* **2006**, *6362*, 63620.
5. Emeis, S.; Schäfer, K. Remote sensing methods to investigate boundary-layer structures relevant to air pollution in cities. *Bound. Layer Meteorol.* **2006**, *121*, 377–385. [CrossRef]
6. Muñoz, R.C.; Undurraga, A.A. Daytime mixed layer over the Santiago basin: Description of two years of observations with a lidar ceilometer. *J. Appl. Meteorol. Climatol.* **2010**, *49*, 1728–1741. [CrossRef]
7. Eresmaa, N.; Härkönen, J.; Joffre, S.M.; Schultz, D.M.; Karppinen, A.; Kukkonen, J. A three-step method for estimating the mixing height using ceilometer data from the Helsinki testbed. *J. Appl. Meteorol. Climatol.* **2012**, *51*, 2172–2187. [CrossRef]
8. Haeffelin, M.; Laffineur, Q.; Bravo-Aranda, J.-A.; Drouin, M.-A.; Casquero-Vera, J.-A.; Dupont, J.-C.; de Backer, H. Radiation fog formation alerts using attenuated backscatter power from automatic lidars and ceilometers. *Atmos. Meas. Tech.* **2016**, *9*, 5347–5365. [CrossRef]
9. Cazorla, A.; Casquero-Vera, J.A.; Román, R.; Guerrero-Rascado, J.L.; Toledano, C.; Cachorro, V.E.; Orza, J.A.G.; Cancillo, M.L.; Serrano, A.; Titos, G.; et al. Near-real-time processing of a ceilometer network assisted with sun-photometer data: Monitoring a dust outbreak over the Iberian Peninsula. *Atmos. Chem. Phys.* **2017**, *17*, 11861–11876. [CrossRef]
10. Chan, K.L.; Wiegner, M.; Flentje, H.; Mattis, I.; Wagner, F.; Gasteiger, J.; Geiß, A. Evaluation of ECMWF-IFS (version 41R1) operational model forecasts of aerosol transport by using ceilometer network measurements. *Geosci. Model Dev.* **2018**, *11*, 3807–3831. [CrossRef]
11. Lee, S.; Hwang, S.-O.; Kim, J.; Ahn, M.-H. Characteristics of cloud occurrence using ceilometer measurements and its relationship to precipitation over Seoul. *Atmos. Res.* **2018**, *201*, 46–57. [CrossRef]
12. Seo, J.-H.; Lee, Y.H.; Kim, Y.-H. Feature selection for very short-term heavy rainfall prediction using evolutionary computation. *Adv. Meteorol.* **2014**, *2014*, 203545. [CrossRef]
13. Moon, S.-H.; Kim, Y.-H.; Lee, Y.H.; Moon, B.-R. Application of machine learning to an early warning system for very short-term heavy rainfall. *J. Hydrol.* **2019**, *568*, 1042–1054. [CrossRef]
14. Lee, M.-K.; Moon, S.-H.; Yoon, Y.; Kim, Y.-H.; Moon, B.-R. Detecting anomalies in meteorological data using support vector regression. *Adv. Meteorol.* **2018**, *2018*, 5439256. [CrossRef]
15. Kim, H.-J.; Park, S.M.; Choi, B.J.; Moon, S.-H.; Kim, Y.-H. Spatiotemporal approaches for quality control and error correction of atmospheric data through machine learning. *Comput. Intell. Neurosci.* **2020**, *2020*, 7980434. [CrossRef] [PubMed]
16. Lee, M.-K.; Moon, S.-H.; Kim, Y.-H.; Moon, B.-R. Correcting abnormalities in meteorological data by machine learning. In Proceeding of the IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, USA, 5–8 October 2014; pp. 902–907.
17. Kim, Y.-H.; Ha, J.-H.; Yoon, Y.; Kim, N.-Y.; Im, H.-H.; Sim, S.; Choi, R.K.Y. Improved correction of atmospheric pressure data obtained by smartphones through machine learning. *Comput. Intell. Neurosci.* **2016**, *2016*, 9467878. [CrossRef]
18. Ha, J.-H.; Kim, Y.-H.; Im, H.-H.; Kim, N.-Y.; Sim, S.; Yoon, Y. Error correction of meteorological data obtained with Mini-AWSs based on machine learning. *Adv. Meteorol.* **2018**, *2018*, 7210137. [CrossRef]
19. Moon, S.-H.; Kim, Y.-H. An improved forecast of precipitation type using correlation-based feature selection and multinomial logistic regression. *Atmos. Res.* **2020**, *240*, 104928. [CrossRef]
20. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
21. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
22. Jain, A.K.; Mao, J.; Mohiuddin, K.M. Artificial neural networks: A tutorial. *IEEE Comput.* **1996**, *29*, 31–44.
23. Gardner, M.W.; Dorling, S.R. Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences. *Atmos. Environ.* **1998**, *32*, 2627–2636.
24. Korea Meteorological Administration. Available online: http://www.kma.go.kr (accessed on 10 July 2020).
25. Vaisala, CL51. Available online: https://www.vaisala.com/en/products/instruments-sensors-and-other-measurement-devices/weather-stations-and-sensors/cl51 (accessed on 10 July 2020).
26. Münkel, C.; Roininen, R. Investigation of boundary layer structures with ceilometer using a novel robust algorithm. In Proceedings of the 15th Symposium on Meteorological Observation and Instrumentation, Atlanta, GA, USA, 16–21 January 2010.

27. Ha, J.-H.; Kim, Y.-H.; Lee, Y.H. Applying artificial neural networks for estimation of planetary boundary layer height. *J. Korean Inst. Intell. Syst.* **2017**, *27*, 302–309. (In Korean) [CrossRef]
28. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y. Stacked denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local denOising Criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.
29. Vaisala PWD20. Available online: https://www.vaisala.com/en/products/instruments-sensors-and-other-measurement-devices/weather-stations-and-sensors/pwd10-20w (accessed on 10 July 2020).
30. American Meteorological Society, Fog, Glossary of Meteorology. Available online: http://glossary.ametsoc.org/wiki/fog (accessed on 10 July 2020).
31. Moon, S.-H.; Kim, Y.-H. Forecasting lightning around the Korean Peninsula by postprocessing ECMWF data using SVMs and undersampling. *Atmos. Res.* **2020**, *243*, 105026. [CrossRef]
32. Goutte, C.; Gaussier, E. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. *Lect. Notes Comput. Sci.* **2005**, *3408*, 345–359.
33. Sun, Y.; Kamel, M.S.; Wong, A.K.; Wang, Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognit.* **2007**, *40*, 3358–3378.
34. Liu, X.-Y.; Wu, J.; Zhou, Z.-H. Exploratory undersampling for class-imbalance learning. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2009**, *39*, 539–550.

applied sciences

MDPI

Article

A Machine Learning-Assisted Numerical Predictor for Compressive Strength of Geopolymer Concrete Based on Experimental Data and Sensitivity Analysis

An Thao Huynh [1], Quang Dang Nguyen [2], Qui Lieu Xuan [3,4], Bryan Magee [1], TaeChoong Chung [5], Kiet Tuan Tran [6] and Khoa Tan Nguyen [2,7,*]

[1] Belfast School of Architecture and the Built Environment, Ulster University, Belfast BT37 0QB, UK; huynh-a@ulster.ac.uk (A.T.H.); b.magee@ulster.ac.uk (B.M.)

[2] Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam; nguyendangquang6@duytan.edu.vn

[3] Faculty of Civil Engineering, Ho Chi Minh City University of Technology, 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City 700000, Vietnam; lieuxuanqui@hcmut.edu.vn

[4] Faculty of Civil Engineering, Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City 700000, Vietnam

[5] Department of Computer Science and Engineering, Kyung Hee University, Gyeonggi-do 130-701, Korea; tcchung@khu.ac.kr

[6] Faculty of Civil Engineering, Ho Chi Minh City University of Technology and Education, Ho Chi Minh 700000, Vietnam; ttkiet@hcmute.edu.vn

[7] Faculty of Civil Engineering, Duy Tan University, Da Nang 550000, Vietnam

[*] Correspondence: nguyentankhoa@duytan.edu.vn; Tel.: +84-82-9270589

Received: 8 October 2020; Accepted: 28 October 2020; Published: 31 October 2020

Abstract: Geopolymer concrete offers a favourable alternative to conventional Portland concrete due to its reduced embodied carbon dioxide (CO_2) content. Engineering properties of geopolymer concrete, such as compressive strength, are commonly characterised based on experimental practices requiring large volumes of raw materials, time for sample preparation, and costly equipment. To help address this inefficiency, this study proposes machine learning-assisted numerical methods to predict compressive strength of fly ash-based geopolymer (FAGP) concrete. Methods assessed included artificial neural network (ANN), deep neural network (DNN), and deep residual network (ResNet), based on experimentally collected data. Performance of the proposed approaches were evaluated using various statistical measures including R-squared (R^2), root mean square error (RMSE), and mean absolute percentage error (MAPE). Sensitivity analysis was carried out to identify effects of the following six input variables on the compressive strength of FAGP concrete: sodium hydroxide/sodium silicate ratio, fly ash/aggregate ratio, alkali activator/fly ash ratio, concentration of sodium hydroxide, curing time, and temperature. Fly ash/aggregate ratio was found to significantly affect compressive strength of FAGP concrete. Results obtained indicate that the proposed approaches offer reliable methods for FAGP design and optimisation. Of note was ResNet, which demonstrated the highest R^2 and lowest RMSE and MAPE values.

Keywords: geopolymer concrete; artificial neural network; machine learning; deep neural network; ResNet; compressive strength; fly ash

1. Introduction

Emission of carbon dioxide caused by various sectors, including construction, industrial processes, transport, residential, and agriculture, has emerged as a severe problem that dramatically affects global climate change. Calcining limestone in Portland cement production represents 8% of global anthropogenic CO_2 emission [1]. Global production of cement increased rapidly from 1.5 billion tonnes in 1998 [2] to 4.1 billion tonnes in 2018 [3], which has significantly impacted emissions linked to the construction sector. This justifies the need for more sustainable alternatives sourced from industrial by-products/wastes with minimal embodied carbon, offering a balance of technical, environmental, and economic benefits.

In this context, geopolymer concrete using industrial by-products (e.g., fly ash and ground-granulated blast-furnace slag) has been reported to reduce up to 80% of CO_2 emission relative to conventional concrete [4]. Geopolymer products are synthesised through the reaction of alkali liquid with silica and alumina contained in aluminosilicate precursors. Depending upon local resources, solid aluminosilicate precursors can be sourced from industrial by-products such as fly ash, metakaolin, red mud, and waste glass [5–8]. According to previous studies [9–13], fly ash-based geopolymer (FAGP) concrete showed the ability to achieve high compressive strengths up to 68 MPa. To assess the compressive strength of concrete, universal compression testing machines are typically used to apply compression load on cylindrical or cube specimens at a prescribed rate (e.g., 20–50 psi/s or 0.14–0.35 MPa/s based on American Society for Testing and Materials (ASTM) C39/C39M-18 [14]). In addition to direct testing, non-destructive testing methods, such as ultrasonic pulse velocity and rebound hammer, are also used to predict the compressive strength of concrete products [15–17]. However, these experimental methods rely heavily on costly equipment and time-consuming preparation of specimens.

As such, artificial intelligence approaches including artificial neural network, adaptive neuro fuzzy inference, and deep learning have been employed to predict the mechanical properties of FAGP concrete by several researchers [18–21]. Inspired by the biological neural system, the artificial neural network (ANN) algorithm with three neuron layers has been widely applied in different research fields such as civil engineering, biochemistry, pharmaceutics, and biology owing to its ability to learn complex relationships among values in its training patterns. Dao et al. [19] investigated the compressive strength of FAGP concrete consisting of steel slag aggregates using ANN and neuro fuzzy inference approaches. Mean absolute error, R-squared, and root mean square error (RMSE) were employed to evaluate the performance of the proposed approaches. Three input parameters including sodium hydroxide (NaOH) concentration, alkali activator/fly ash ratio, and sodium hydroxide-to-sodium silicate ratio were used to predict the compressive strength of FAGP concrete. Results obtained from the ANN were in substantial agreement with experiment data. Sensitivity analysis for ANN and adaptive neuro fuzzy inference were adopted in a study by the same authors [18] to evaluate the impact of each input factor including the mass of fly ash, sodium silicate (Na_2SiO_3), NaOH, and water on the accuracy of the proposed models. The two approaches effectively predicted the compressive strength of the geopolymer concrete using only three input parameters in the mixture proportion. Curing condition factors were neglected in these studies even though they undoubtedly play essential roles in the compressive strength of geopolymer concrete [22–25]. In addition to mixture proportion factors, curing time and temperature values were added in the training dataset for compressive strength prediction of FAGP concrete using ANN in a study by Ling et al. [21]. Results from ANN modelling methods showed good agreement with those obtained from experiments. The authors concluded that compressive strength of geopolymer concrete was profoundly influenced by mixture proportion and curing conditions. Performance of deep neural network (DNN) and deep residual network (ResNet) approaches in predicting the compressive strength of FAGC was investigated in a study by Nguyen et al. [20]. With high rate of recognition accuracy within a complex network, the ResNet model showed better performance than the DNN models, with two main forward and backward passes; therefore, it has been used in several advanced engineering problems [26,27]. ResNet and DNN approaches were also employed to predict compressive strength of conventional and foamed concrete in the studies by Jang et al. [28] and Nguyen et al. [29], respectively. Against this background, current solutions to predict compressive strength of FAGP concrete have not been dealt with in depth within existing literature. Although various machine learning approaches including ANN and

DNN have been separately introduced in several studies [19,21] as numerical predictors for FAGP strength, a thorough search of relevant published literature yielded a mere presence of the ResNet approach in FAGP property prediction. The lack of studies on impacts of input parameters (e.g., mix proportion ratios, NaOH concentration, and curing conditions) on geopolymer strength indicates possible improvements for upcoming research. More comprehensive research needs to be carried out to investigate the effectiveness of various machine learning methods in predicting compressive strength of FAGP concrete, considering a wider variety of input parameters and sensitivity analysis.

As such, this study aims to offer advancements to the existing literature by employing ANN, DNN, and ResNet approaches integrated with sensitivity analysis to predict the compressive strength of FAGP concrete. These models were trained through 263 pairs of input/target values obtained from experiments. Performance of FAGP strength prediction of the three proposed approaches was investigated in two phases. In the first phase, the models were trained and validated using randomly shuffled datasets. Additional training and assessment under K-fold cross validation schemes were then carried out to confirm the results obtained from the first phase. Impacts of six input parameters (including NaOH/Na_2SiO_3 ratio, fly ash/aggregate ratio, alkali liquid/fly ash ratio, NaOH concentration, curing time, and temperature) on prediction models were investigated using sensitivity analysis. Outcomes from sensitivity analysis are expected to identify the critical input parameters in FAGP strength prediction and control them carefully during geopolymer production. Three measures including R-squared (R^2), root mean square error (RMSE), and mean absolute percentage error (MAPE) were employed to evaluate the accuracy of the proposed machine learning techniques.

2. Machine Learning Approaches

2.1. Artificial Neural Network (ANN)

Inspired by the biological neuron system, ANN is based on a suite of mutually connected units, known as perceptrons, which replicate the functions of neurons in the human brain. ANN is one of the main models used in machine learning where its structure is formed by three layers of neurons including input, hidden, and output layers. Independent variables enter the system through the input layer and are processed in the hidden layer, while predicted values are generated in the output layer. Figure 1 presents the basic concept of ANN.

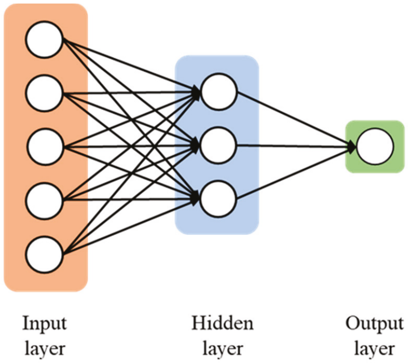


Figure 1. The construction of the artificial neural network (ANN) [20].

2.2. Deep Neural Network (DNN)

DNN consists of more layers and neurons than ANN, leading to its ability to learn functions with a high degree of complexity. DNN possesses a powerful representational ability of input data and can reduce over-fitting issues in regression performance [30]. With powerful representational ability, DNN is able to achieve high accuracy in various tasks [31]. A typical DNN network structure is presented in Figure 2, including two main forward and backward phases.

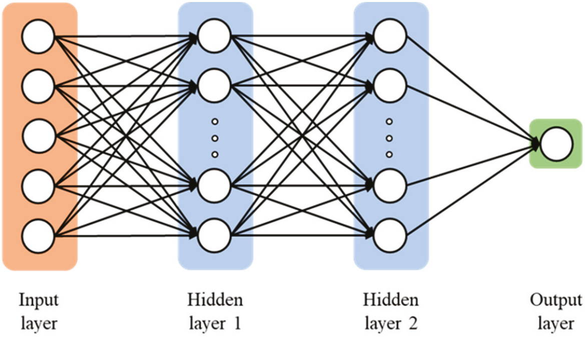


Figure 2. Deep neural network (DNN) with two hidden layers [20].

2.3. Deep Residual Network (ResNet)

ResNet was developed to overcome a limitation in training deep networks where training errors can increase as the number of layers increases [20]. Owing to modified architectures, ResNet models have been empirically confirmed to enhance learnability of neural networks with less error on defined tasks using a limited number of layers [32]. ResNet consists of residual blocks with shortcut connections as shown in Figure 3, where the formulation H(x) is the desired mapping output of a specific layer and x is the input data. Given the presence of shortcut connections, gradient-based optimisation algorithms work effectively under ResNet-based architectures and improve the learnability of weight layers representing the function F(x) [33].

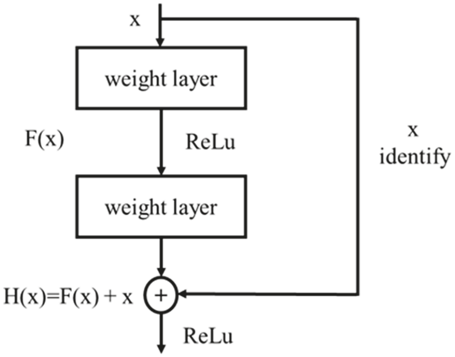


Figure 3. A block in a deep residual network [20].

3. Experimental Programme

3.1. Materials and Mixing Process

Constituent materials of the FAGP concretes considered were fly ash, coarse and fine aggregates, alkali activator, and water. Low-calcium fly ash (class F) with a specific gravity of 2500 kg/m^3 was used as the main aluminosilicate precursor. The chemical composition of the fly ash used is presented in Table 1, which conforms to requirements from ASTM 618 [ASTM]. The FAGP concrete mix designs and mixing processes were based on a previous study by Nguyen et al. [20]. Geopolymer mix designs were formulated based on various binder and aggregate contents, concentration of sodium hydroxide, and curing conditions. The ratio of fly ash mass to total aggregate mass (fly ash/aggregate) varied from 0.13–0.37. Specific gravities of the coarse and fine aggregates were 2700 kg/m^3 and 2650 kg/m^3, respectively.

Table 1. Chemical compositions of fly ash class F.

Oxide	SiO_2	Al_2O_3	Fe_2O_3	CaO	K_2O & Na_2O	MgO	SO_3	LOI
(%)	51.7	31.9	3.48	1.21	1.02	0.81	0.25	9.63

Sodium silicate solution consisting of 36% Na_2O and 38% SiO_2 by mass was mixed with sodium hydroxide with a wide range of concentrations including 4M, 8M, 11M, 12M, 15M, and 18M to prepare alkali liquid (AL). The ratios of NaOH/Na_2SiO_3 and AL/fly ash ranged from 0.4–2.5 and 0.3–0.7, respectively.

Fly ash and aggregates were mixed together on a slow setting for about three minutes. Alkali solution was then added and mixed for a further four minutes before casting. Fresh FAGP concrete was cast in standard cylinder moulds (100 mm diameter, 200 mm high), de-moulded after 24 h, and then cured in an oven at temperatures 40, 60, 80, 90, 100, and 120 °C for 2, 4, 6, 8, 10, and 12 h. The processing and testing procedure is represented in Figure 4.

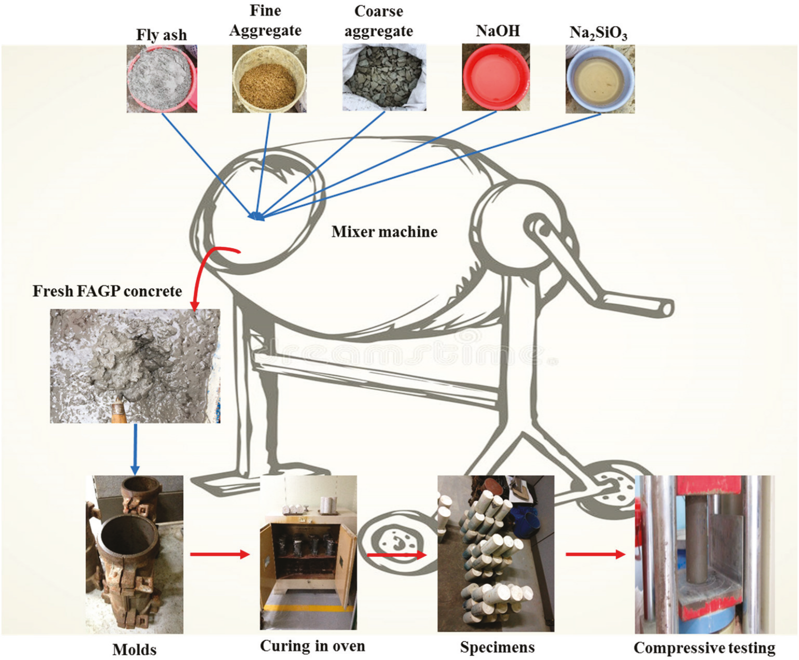


Figure 4. Schematic illustration of experimental works.

3.2. Data Preparation for Machine Learning Approaches

According to previous studies [10,22,34], FAGP concrete properties depend on constituent material proportioning, concentration of sodium hydroxide (C_M), and curing conditions. In this study, a total of 263 pairs of input/target values fabricated from different geopolymer mix proportions, NaOH concentration, and curing conditions were designed to generate the data for running the machine learning-based models. Inside these models, the six input variables considered to estimate the compressive strength of FAGP concrete were: NaOH/Na_2SiO_3, fly ash/aggregate and AL/fly ash, C_M, curing time, and curing temperature. For compressive strength measurement, FAGP concrete cylinders were subjected to axial compression with a loading rate of up to 0.35 MPa/s according to ASTM C39/C 39M-18 [14] after seven days. At least three specimens were tested for each mix design of FAGP concrete to obtain the mean value of the compressive strength. The test data from experimental works are given in Table 2.

Table 2. Statistical parameters of fly ash-based geopolymer (FAGP) concrete used in the training dataset.

Parameter	Unit	Value	Variable
NaOH/Na_2SiO_3		Min. 0.4–Max. 2.5	Input
Fly ash/Aggregate		Min. 0.13–Max. 0.37	
AL/Fly ash		Min. 0.3–Max. 0.7	
NaOH concentration	M	4, 8, 11, 12, 15, 18	
Curing time	h	2, 4, 6, 8, 10	
Curing temperature	°C	40, 60, 80, 90, 100, 120	
Compressive strength	MPa	5.44–67.86	Output
Total number of datasets		263	

4. Research Methodology

In this study, 263 datasets (each comprising six inputs and one output) were used to train and validate ANN, DNN, and ResNet models. In terms of inputs, each dataset comprised a unique combination of the six mix design values considered, as summarised in Table 2. The output for each dataset was the corresponding average compressive 7-day strength result obtained from experimental testing. The range of strength values recorded for the 263 combinations considered was 5.55–67.86 MPa.

A data division scheme was applied to reduce possibilities of error and improve the reliability of predicted results. Random selection of about 90% of the values (235 datasets) in the training dataset were chosen from the original data collection to train the network, while the remaining values (28 datasets) remained untrained as a validation database to confirm the accuracy of the trained network. The structures of three machine learning approaches including ANN, DNN, and ResNet are presented in the schematic flowchart in Figure 5.

FAGP concrete compressive strength was predicted by employing ANN, DNN, and ResNet architectures comprising weight, normalisation, and activation layers in regression tasks. For comparative purposes, DNN and ResNet models consisted of the same number of nodes with 128 nodes in Weight Layer 1 and 256 nodes in Weight Layer 2. A layer with 256 nodes, known as Weight Layer 3, was included to enable additional operation at the end of ResNet implementation. The ANN model with one weight layer comprised 384 nodes. One of the stochastic gradient descent methods, known as Adam [35], was used as the optimisation method to update neural networks coefficients since it integrated advanced features from different optimisation algorithms, including AdaGrad and RMSProp. The layer normalisation method introduced by Ba et al. [36] was employed to ensure inputs to layers fell within specific ranges since it exhibited efficient training time in neural network architecture compared to traditional batch normalisation. Training models without normalisation were also carried out to validate the effectiveness of the model integrated with layer normalisation. During the training process, dropping out units with keep probability of 0.2 in the architectures were included in the final models to prevent overfitting problems. Table 3 presents details of the setting of six architectures (known as architectures 1–6) implemented in this study.

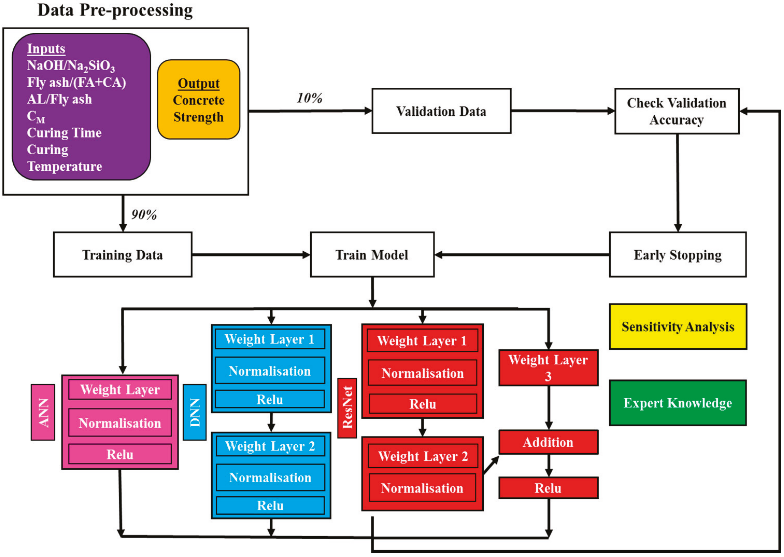


Figure 5. Schematic flowchart presenting the three machine learning approaches used to estimate FAGP concrete compressive strength.

Table 3. Details of the setting of six investigated architectures from artificial neural network (ANN), deep neural network (DNN) and deep residual network (ResNet).

Properties	Architectures					
	1	2	3	4	5	6
Network	ANN	ANN	DNN	DNN	ResNet	ResNet
Layer Normalisation	Yes	No	Yes	No	Yes	No
Activation Function Rectified linear unit (ReLU)	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
Drop-out	Yes	Yes	Yes	Yes	Yes	Yes

Three statistical measures including R^2, RMSE, and MAPE were applied to evaluate the accuracy of the proposed machine learning approaches under the K-fold cross validation scheme. These parameters provide insights into differences between original and estimated values. Higher R^2 value and/or lower MAPE and RMSE values indicate better prediction performance of machine learning approaches [19]. The three statistical measures were calculated using the following equations:

$$R^2 = \frac{\left(n\sum_i y_i y_i' - \sum_i y_i' \sum_i y_i\right)^2}{\left(n\sum_i y_i'^2 - \left(\sum_i y_i'\right)^2\left(n\sum_i y_i^2 - \left(\sum_i y_i\right)^2\right)\right)}, \tag{1}$$

$$MAPE = \frac{1}{n}\sum\left|\frac{y_j - y_j'}{y_j}\right| \times 100, \tag{2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j - y_j'\right)^2}, \tag{3}$$

where y_j and y_j' are the compressive strength obtained from experiments and predictions respectively; n is the number of datasets.

The K-fold cross validation method divides data into K equal folds and then does K independent training iterations on the prediction model with ($K - 1$) folds while leaving the remaining fold for validation purposes. In this experiment, the common value $K = 10$ was used. The performance of the prediction model was judged by averaging the metric measurement (R^2, MAPE, and RMSE) measured in K training and evaluating the iterations as follows:

$$M_{K-fold} = \frac{1}{K}\sum_{k=1}^{K} m_k, \tag{4}$$

where $M_{K\text{-}fold}$ denotes a general metric measurement when K-fold cross validation is applied, and m_k is the metric measurement in the fold k of the procedure.

An important note is that the same training and validation sets in each fold were used to train and validate each model. A hypothesis test (e.g., paired t-test) with a significance level $\alpha = 0.05$ was then applied to the accurate measurements of each model on validation sets in 10 divided folds to confirm the statistical significance of the results. The null hypothesis was that these measurements are all in the same population (or belong to the same model), suggesting there is no difference between the performance of two evaluated models. From the t-test, a p-value less than the chosen significance level ($\alpha = 0.05$) can statistically confirm the advance of a prediction model over the others (rejecting the null hypothesis), while the p-value greater than this significance level may suggest that the event, or the numerical conclusion, happens by chance (not rejecting the null hypothesis).

5. Experimental Programme

5.1. Estimative Performance of ANN, DNN, and ResNet Approaches

In the first phase, six predictive models based on three proposed machine learning approaches (ANN, DNN, and ResNet) were trained and validated using randomly shuffled datasets obtained from experimental works. Input variables in the dataset consisted of six parameters including mixture proportions (i.e., $NaOH/Na_2SiO_3$, fly ash/aggregate, AL/fly ash), NaOH concentration, and curing conditions. Compressive strength of FAGP specimens was regarded as output variable. The results from the first phase were aimed to provide a short list of models to further test with K-fold cross validation and the t-test method as described in Section 4.

R^2, RMSE, and MAPE values for the ANN (architecture 1 and 2), DNN (architecture 3 to 4), and ResNet (architecture 5 to 6) models are summarised in Table 4, with the bold numbers representing the best predictive model of each approach. As shown, architectures 1, 3, and 6 were found to be the best ANN, DNN, and ResNet models, respectively. From the six architectures presented in Table 4, ResNet-based architecture 6 was the best model for determining FAGP concrete compressive strength with the highest R^2 of 0.937 and lowest RMSE and MAPE values (1.987 and 6.6, respectively). Apart from the ResNet models, ANN-based architecture 1 (R^2 = 0.889; RMSE = 4.711; MAPE = 14.06) and DNN-based architecture 3 (R^2 = 0.898; RMSE = 2.521; MAPE = 9.496) showed better predictive performance than the other models (architecture 2, 4, and 5). Based on these observations, ANN-based architecture 1, DNN-based architecture 3, and ResNet-based architecture 6 were selected for further investigation.

Table 4. Performance comparison of six architectures for FAGP compressive strength prediction in terms of R-squared (R^2), root mean square error (RMSE) and mean absolute percentage error (MAPE).

Models	**Metric**	**Training**			**Validation**		
		R^2	**RMSE**	**MAPE**	**R^2**	**RMSE**	**MAPE**
ANN	Architecture 1	**0.921**	**3.153**	**9.291**	**0.889**	**4.711**	**14.06**
	Architecture 2	0.827	4.802	12.712	0.798	5.059	11.818
DNN	Architecture 3	**0.923**	**3.273**	**11.54**	**0.898**	**2.521**	**9.496**
	Architecture 4	0.821	5.004	14.03	0.84	3.161	13.486
ResNet	Architecture 5	0.921	3.244	9.105	0.915	3.288	8.581
	Architecture 6	**0.896**	**3.815**	**9.797**	**0.937**	**1.987**	**6.600**

In the second phase, further investigation into performance of the proposed approaches was carried out using additional training and assessment under a 10-fold scheme with three architectures: 1, 3, and 6. Results of various statistic measures (R^2, MAPE, and RMSE) for each fold and the average (Avg.) values with standard deviations are presented in Table 5. The same training and validation sets of each fold were applied for the three models 1, 3, and 6. As shown in this table, ResNet-based architecture 6 obtained the best strength prediction performance in terms of R^2 (0.934 ± 0.021), RMSE (2.750 ± 0.573), and MAPE (8.552 ± 1.333. Also, a further paired *t*-test with $\alpha = 0.05$ was applied to prove the statistical significance of this observation. As presented in Table 6, *p*-values from the comparisons of ResNet model and ANN/DNN models were lower than the chosen significance level ($\alpha = 0.05$), providing statistical confirmation that the ResNet model out-performed the ANN/DNN model in terms of FAGP strength prediction.

Relationships between the experimental and predicted strength values from architectures 1, 3, and 6 are illustrated in Figure 6. As shown, compressive strength values predicted by all machine learning models were close to the actual values obtained from compression experiments, indicating that the proposed approaches were successfully trained to predict FAGP compressive strength. The ResNet model outperformed the other models with the strongest relationship existing between actual and predicted values. This observation was confirmed in Figure 7, which presents the correlation coefficient (R) of the three approaches in terms of validation data. Minimal variation existed between actual and predicted values existed for the ANN, DNN, and ResNet models, albeit with the highest variance being associated with the former (architecture 1).

The relationship between iterations of the three best performed architectures (1, 3, and 6) and validation RMSE is shown in Figure 8. The highest convergence speed was observed in ResNet-based architecture 6 model, which required only 2000 iterations to reach a validation RMSE of 4.8 MPa. For the same RMSE, higher iteration numbers of 6000 and 148,000 were required for DNN and ANN models, respectively. After convergence, sufficiently low values of RMSE were observed in the ResNet and DNN models, indicating better performances over the ANN model. For instance, at the same iteration value of 152,000, the ANN model converged at an RMSE of 4.7 MPa while ResNet and DNN models achieved lower values of RMSE (2.1 MPa and 2.7 MPa, respectively).

Table 5. Accuracy measurements on validation sets for the proposed machine learning approaches (architectures 1, 3, and 6) under K-fold cross validation scheme.

	Fold	R^2	RMSE	MAPE
ANN	1	0.830	4.650	15.807
	2	0.894	3.447	9.696
	3	0.899	3.954	10.676
	4	0.888	3.498	10.354
	5	0.937	2.805	7.440
	6	0.914	4.174	8.366
	7	0.873	4.080	9.658
	8	0.927	2.367	7.728
	9	0.943	2.615	7.189
	10	0.832	4.409	13.452
	Avg.	**0.893 ± 0.038**	**3.600 ± 0.748**	**10.036 ± 2.621**
DNN	1	0.908	3.218	9.398
	2	0.915	3.074	10.047
	3	0.907	3.812	11.298
	4	0.910	3.134	8.830
	5	0.937	2.803	9.663
	6	0.879	4.947	9.030
	7	0.910	3.436	8.948
	8	0.937	2.988	7.876
	9	0.946	2.542	6.515
	10	0.866	3.951	13.264
	Avg.	**0.912 ± 0.023**	**3.391 ± 0.659**	**9.487 ± 1.740**
ResNet	1	0.929	1.907	9.043
	2	0.923	3.097	9.130
	3	0.921	3.512	9.919
	4	0.946	2.441	7.738
	5	0.950	2.504	7.692
	6	0.926	3.367	7.654
	7	0.939	2.815	7.831
	8	0.967	1.929	7.406
	9	0.950	2.434	7.400
	10	0.886	3.495	11.709
	Avg.	**0.934 ± 0.021**	**2.750 ± 0.573**	**8.552 ± 1.333**

Table 6. Paired *t*-test for statistical significance between ResNet and ANN-based models (architecture 6 and 1) and between ResNet and DNN-based models (architecture 6 and architecture 3).

***p*-Value**	**ResNet vs. ANN**	**ResNet vs. DNN**
R^2	0.00193	0.00053
RMSE	0.00616	0.00395
MAPE	0.04952	0.00499

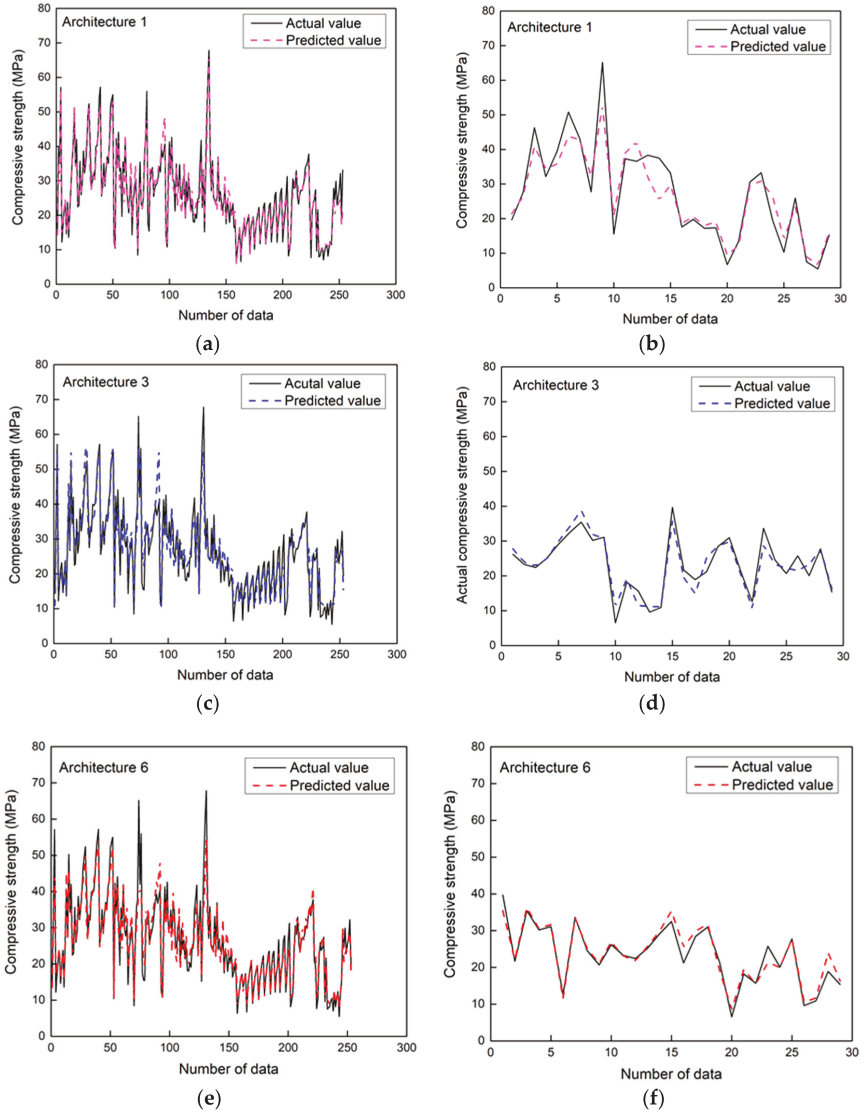
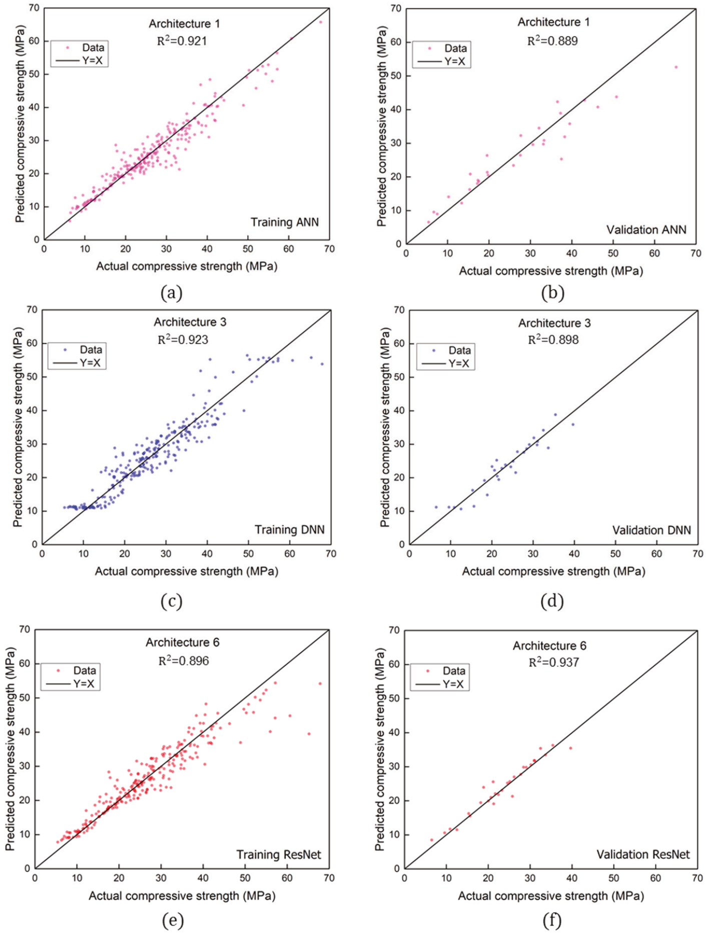


Figure 6. Relationship between compressive strength values obtained from experiments (actual value) and machine learning approaches (predicted value): (**a**) training ANN; (**b**) validation ANN; (**c**) training DNN; (**d**) validation DNN; (**e**) training ResNet; (**f**) validation ResNet.



(a) (b)

(c) (d)

(e) (f)

Figure 7. Correlation coefficients R of three proposed approaches: (**a**) training ANN; (**b**) validation ANN; (**c**) training DNN; (**d**) validation DNN; (**e**) training ResNet; (**f**) validation ResNet.

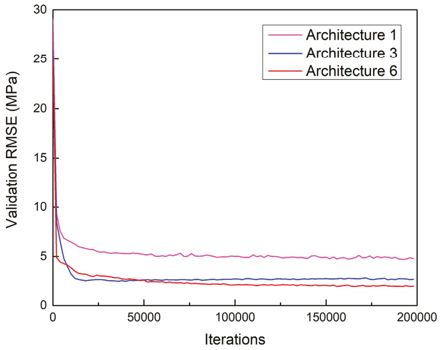


Figure 8. Relationship between validation RMSE and iterations.

Figure 9 presents the distribution of error rates at 5% increments for predicted results obtained from architectures 1, 3, and 6. It is noted that the majority of datasets (61%) from the ResNet model exhibited error levels less than 5%. Corresponding frequencies of errors less than 5% for the ANN and DNN models were significantly lower (approximately 21 and 35%, respectively). In terms of errors less than 20%, frequencies for the ANN, DNN, and ResNet models were 79, 89, and 89%, respectively. In terms of ranking, therefore, the ResNet model provided the best estimative performance, followed by the DNN and ANN models.

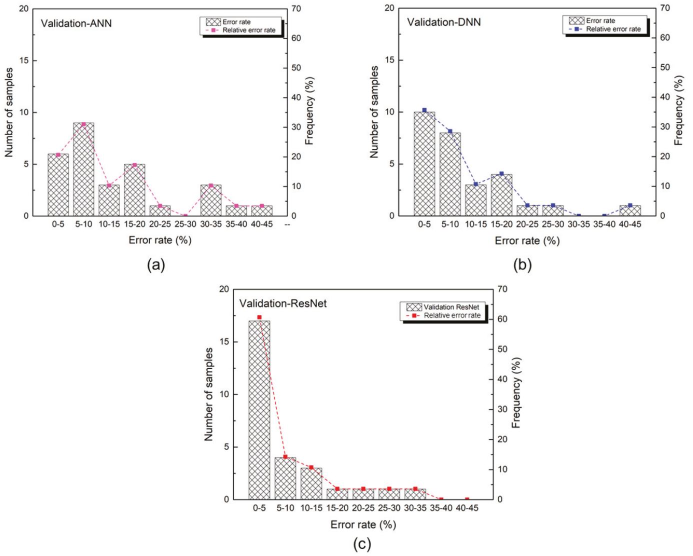


Figure 9. Error rate distribution of three proposed approaches: (**a**) validation ANN; (**b**) validation DNN; (**c**) validation ResNet.

5.2. Sensitivity Analysis

Sensitivity analysis is commonly used to evaluate how input parameters affect output variation derived by machine learning models [37]. As the best performing model, ResNet-based architecture 6 was exclusively selected for this analysis, which involved calculating FAGP concrete compressive strength by changing one input variable at a time while maintaining the other five as constants based on their mean values. For example, to assess the importance of the NaOH/Na_2SiO_3 ratio, this value was varied from 0.4–2.5, while fly ash/aggregate, AL/Fly ash, NaOH concentration, curing time, and temperature values were kept constant at mean values of 0.23, 0.5, 14 (M), 8 h, and 85.6 °C, respectively. Data derived from this sensitivity analysis were returned to the training process to estimate compressive strength. For each parameter, a corresponding sensitivity analysis factor was given by the expression:

$$I_i = f_{max}(x_i) - f_{min}(x_i), \tag{5}$$

$$SA_i = \frac{I_i}{\sum\limits_i I_i} \times 100, \tag{6}$$

where $f_{max}(x_i)$ and $f_{min}(x_i)$ are the maximum and minimum estimated compressive strengths relating to the input variable x_i, with all other input parameters kept constant at their mean values.

Figure 10 shows the results of this sensitivity analysis, from which a pronounced influence (35.5%) of fly ash/aggregate ratio on estimated compressive strength can be seen. A similar effect was observed in the study by Joseph and Mathew [11], and can be explained by the fact that the internal void structure formed by fly ash and aggregates has direct effects on FAGP compressive strength. Additionally, shown in this figure are high sensitivity scores of 16.22%, 16.18%, and 14.93% for curing time, NaOH concentration, and curing temperature, respectively. This confirms the observations from previous studies by [22,23] where curing conditions play significant roles in compressive strength of FAGP concrete. As such, various factors such as mix proportions, sodium hydroxide concentration, and curing regimes should be thoroughly considered in the prediction of FAGP mechanical properties using machine learning approaches. In particular, based on these findings, it is recommended that the fly ash/aggregate ratio is carefully determined and controlled in geopolymer manufacturing processes owing to its pronounced effect on FAGP strength.

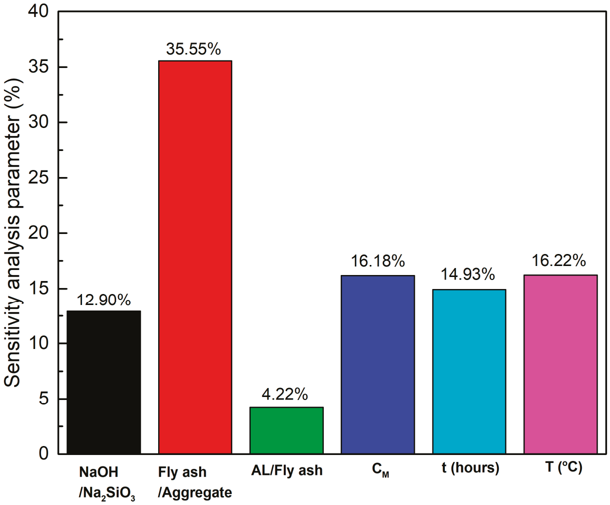


Figure 10. Sensitivity analysis parameters for the estimated compressive strength of FAGP concrete.

6. Conclusions

This study employed three different machine learning approaches including ANN, DNN, and ResNet to predict compressive strength of fly ash-based geopolymer concrete. Six parameters of mix design and curing conditions (including NaOH/Na$_2$SiO$_3$, fly ash/aggregate, AL/Fly ash, concentration of sodium hydroxide, curing time, and temperature) and corresponding 7-day compressive strength results were used to generate 263 unique input/output pairs for model training purposes.

While the results indicated that all three machine learning approaches could predict FAGP concrete compressive strength with some degree of accuracy, the ResNet model was the most promising method with the highest R^2 (0.937) and the lowest RMSE (1.987) and MAPE (6.6) values. This observation was confirmed by additional training and assessment under the K-fold cross validation scheme and paired *t*-test with $\alpha = 0.05$, where the highest R^2 (0.934 ± 0.021) and the lowest RMSE (2.750 ± 0.573) and MAPE (8.552 ± 1.333) were observed in the ResNet-based model. Sensitivity analysis performed for the ResNet model confirmed that the ratio of fly ash/aggregate was the most dominant factor when predicting compressive strength, with a sensitivity analysis score of 35%. This was followed in order of importance by curing temperature (16.22%), NaOH concentration (16.18%), curing time (14.93%), NaOH/Na$_2$SiO$_3$ ratio (12.90%), and AL/fly ash ratio (4.22%). This analysis indicates the importance of considering a wide range of input parameters in the prediction of FAGP concrete compressive strength and controlling them carefully during the manufacturing process.

This study provides a detailed understanding of performance of different machine learning approaches in strength prediction for FAGP concrete. The findings highlight potential uses of the proposed machine learning approaches such as ResNet and DNN as effective tools to, not only precisely predict mechanical properties of FAGP, but also to develop mix designs for geopolymer concrete. In the next phase of work, consideration will be given to how ResNet and DNN models can be applied in FAGP manufacturing industries to predict optimised mix designs and curing regimes based on target compressive strength. This predictive ability will also be linked to mix design evaluations in terms of potential cost and environmental benefits prior to construction stages. In addition, the proposed machine learning approaches adopted a general training scheme for neural networks with standard input and output features, indicating promising potential to be applied to other regression problems in upcoming research works.

Author Contributions: Conceptualization, Q.D.N. and K.T.N.; Data curation, Q.D.N. and Q.L.X.; Funding acquisition, T.C.; Investigation, A.T.H., Q.D.N. and K.T.N.; Methodology, Q.D.N. and Q.L.X.; Project administration, K.T.N.; Resources, K.T.N.; Software, Q.D.N. and Q.L.X.; Supervision, B.M. and T.C.; Validation, Q.L.X. and T.C.; Visualization, K.T.N.; Writing—original draft, A.T.H. and K.T.N.; Writing—review & editing, A.T.H., Q.L.X., K.T.T. and B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Basic Science Research Program through the National Research Foundation of Korea (NRF-2020R1F1A1050014).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Farfan, J.; Fasihi, M.; Breyer, C. Trends in the global cement industry and opportunities for long-term sustainable CCU potential for Power-to-X. *J. Clean. Prod.* **2019**, *217*, 821–835. [CrossRef]
2. van Oss, H.G. *Mineral Commodity Summaries: Cement*; U.S. Geological Survey: Reston, VA, USA, 2000.
3. van Oss, H.G. *Mineral Commodity Summaries: Cement*; U.S. Geological Survey: Reston, VA, USA, 2018.
4. McGrath, T.E.; Kwasny, J.; Aiken, T.; Cox, S.; Soutsos, M.; Chen, J.F.; Mariotti, J.; Sha, W.; Correia, R. Demonstration of using low carbon precast concrete products for an energy efficient built environment. *Sustain. Constr. Mater. Technol.* **2019**. [CrossRef]
5. Meesala, C.R.; Verma, N.K.; Kumar, S. Critical review on fly-ash based geopolymer concrete. *Struct. Concr.* **2020**, *21*, 1013–1028. [CrossRef]
6. Luukkonen, T.; Abdollahnejad, Z.; Yliniemi, J.; Kinnunen, P.; Illikainen, M. One-part alkali-activated materials: A review. *Cem. Concr. Res.* **2018**, *103*, 21–34. [CrossRef]
7. Vafaei, M.; Allahverdi, A. High strength geopolymer binder based on waste-glass powder. *Adv. Powder Technol.* **2017**, *28*, 215–222. [CrossRef]
8. Detphan, S.; Chindaprasirt, P. Preparation of fly ash and rice husk ash geopolymer. *Int. J. Miner. Metall. Mater.* **2009**, *16*, 720–726.
9. Guru Jawahar, J.; Lavanya, D.; Sashidhar, C. Performance of fly ash and ggbs based geopolymer concrete in acid environment. *Int. J. Res. Sci. Innov.* **2016**, *3*, 101–104.
10. Al Bakri, A.M.M.; Kamarudin, H.; Bnhussain, M.; Rafiza, A.R.; Zarina, Y. Effect of Na_2SiO_3/NaOH Ratios and NaOH Molarities on Compressive Strength of Fly-Ash-Based Geopolymer. *ACI Mater. J.* **2012**, *109*, 503–508.
11. Joseph, B.; Mathew, G. Influence of aggregate content on the behavior of fly ash based geopolymer concrete. *Sci. Iran.* **2012**, *19*, 1188–1194. [CrossRef]
12. Farhan, N.A.; Sheikh, M.N.; Hadi, M.N. Investigation of engineering properties of normal and high strength fly ash based geopolymer and alkali-activated slag concrete compared to ordinary Portland cement concrete. *Constr. Build. Mater.* **2019**, *196*, 26–42. [CrossRef]
13. Tempest, B.; Sanusi, O.; Gergely, J.; Ogunro, V.; Weggel, D. Compressive strength and embodied energy optimization of fly ash based geopolymer concrete. In Proceedings of the 3rd World Coal Ash, WOCA Conference, Lexington, KY, USA, 4–7 May 2009.
14. American Society for Testing and Materials. *Standard Test Method for Compressive Strength of Cylindrical Concrete Specimens*; ASTM C39/C39M-18; ASTM International: West Conshohocken, PA, USA, 2018.
15. Naskar, S.; Chakraborty, A.K. Effect of nano materials in geopolymer concrete. *Perspect. Sci.* **2016**, *8*, 273–275. [CrossRef]
16. Huynh, A.T.; Magee, B.; Woodward, D. A Preliminary Characterisation of Innovative Semi-Flexible Composite Pavement Comprising Geopolymer Grout and Reclaimed Asphalt Planings. *Materials* **2020**, *13*, 3644. [CrossRef]

17. Prabhakaran, A.; Rajagopal, G.; Manikandan, S.A. Non Destructive Testing on Geopolymer Concrete using Concrete Demolition Waste. *SSRG Int. J. Civ. Eng.* **2017**, 610–614.
18. Van Dao, D.; Ly, H.-B.; Trinh, S.H.; Le, T.-T.; Pham, B.T. Artificial Intelligence Approaches for Prediction of Compressive Strength of Geopolymer Concrete. *Materials* **2019**, *12*, 983. [CrossRef]
19. Van Dao, D.; Trinh, S.H.; Ly, H.-B.; Pham, B.T. Prediction of Compressive Strength of Geopolymer Concrete Using Entirely Steel Slag Aggregates: Novel Hybrid Artificial Intelligence Approaches. *Appl. Sci.* **2019**, *9*, 1113. [CrossRef]
20. Nguyen, K.T.; Nguyen, Q.D.; Le, T.A.; Shin, J.; Lee, K. Analyzing the compressive strength of green fly ash based geopolymer concrete using experiment and machine learning approaches. *Constr. Build. Mater.* **2020**, *247*, 118581. [CrossRef]
21. Ling, Y.; Wang, K.; Wang, X.; Li, W. Prediction of engineering properties of fly ash-based geopolymer using artificial neural networks. *Neural Comput. Appl.* **2019**, *3*, 1–21. [CrossRef]
22. Muhammad, N.; Baharom, S.; Amirah, N.; Ghazali, M.; Alias, N.A. Effect of Heat Curing Temperatures on Fly Ash-Based Geopolymer Concrete. *Int. J. Eng. Technol.* **2019**, *8*, 15–19.
23. Patil, A.A.; Chore, H.S.; Dode, P.A. Effect of Curing Conditions on Compressive Strength of Brick Aggregate. *Adv. Concr. Constr.* **2014**, *2*, 29–37. [CrossRef]
24. Zhang, H.; Li, L.; Sarker, P.K.; Long, T.; Shi, X.; Wang, Q.; Cai, G. Investigating Various Factors Affecting the Long-Term Compressive Strength of Heat-Cured Fly Ash Geopolymer Concrete and the Use of Orthogonal Experimental Design Method. *Int. J. Concr. Struct. Mater.* **2019**, *13*, 63. [CrossRef]
25. Azevedo, A.G.S.; Strecker, K.; Barros, L.A.; Tonholo, L.F.; Lombardi, C.T. Effect of Curing Temperature, Activator Solution Composition and Particle Size in Brazilian Fly-Ash Based Geopolymer Production. *Mater. Res.* **2019**, *22*, 1–12. [CrossRef]
26. Nguyen, D.Q.; Vien, N.A.; Dang, V.-H.; Cheong, T. Asynchronous framework with Reptile+ algorithm to meta learn partially observable Markov decision process. *Appl. Intell.* **2020**, *50*, 4050–4062. [CrossRef]
27. Choi, S.; Le Pham, T.; Nguyen, Q.D.; Layek, A.; Lee, S.; Chung, T. Toward Self-Driving Bicycles Using State-of-the-Art Deep Reinforcement Learning Algorithms. *Symmetry* **2019**, *11*, 290. [CrossRef]
28. Jang, Y.; Ahn, Y.; Kim, H.Y. Estimating Compressive Strength of Concrete Using Deep Convolutional Neural Networks with Digital Microscope Images. *J. Comput. Civ. Eng.* **2019**, *33*, 04019018. [CrossRef]
29. Nguyen, T.; Kashani, A.; Ngo, T.; Bordas, S. Deep neural network with high-order neuron for the prediction of foamed concrete strength. *Comput. Civ. Infrastruct. Eng.* **2019**, *34*, 316–332. [CrossRef]
30. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 30 September 2020).
31. Sze, V.; Chen, Y.-H.; Yang, T.-J.; Emer, J.S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. In *Proceedings of the IEEE*; IEEE: New York, NY, USA, 2017; Volume 105, pp. 2295–2329.
32. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
33. Wang, R.; Chencho; An, S.; Li, J.; Li, L.; Hao, H.; Liu, W. Deep residual network framework for structural health monitoring. *Struct. Health Monit.* **2020**, 1–19. [CrossRef]
34. Fauzi, A.; Nuruddin, M.F.; Malkawi, A.B.; Abdullah, M.M.A.B.; Mohammed, B.S. Effect of Alkaline Solution to Fly Ash Ratio on Geopolymer Mortar Properties. *Key Eng. Mater.* **2017**, *733*, 85–88. [CrossRef]
35. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015.
36. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
37. Tenza-Abril, A.; Villacampa, Y.; Solak, A.; Baeza-Brotons, F. Prediction and sensitivity analysis of compressive strength in segregated lightweight concrete based on artificial neural network using ultrasonic pulse velocity. *Constr. Build. Mater.* **2018**, *189*, 1173–1183. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Structural Vibration Tests: Use of Artificial Neural Networks for Live Prediction of Structural Stress

Laura Wilmes [1], Raymond Olympio [2], Kristin M. de Payrebrune [1,*] and Markus Schatz [2]

[1] Institute for Computational Physics in Engineering, TU Kaiserslautern, 67663 Kaiserslautern, Germany; laura@wilmes.lu

[2] AIRBUS Defense & Space GmbH, 88090 Immenstaad, Germany; raymond.olympio@airbus.com (R.O.); markus.schatz@airbus.com (M.S.)

* Correspondence: kristin.payrebrune@mv.uni-kl.de

Received: 31 October 2020; Accepted: 26 November 2020; Published: 29 November 2020

Abstract: One of the ongoing tasks in space structure testing is the vibration test, in which a given structure is mounted onto a shaker and excited by a certain input load on a given frequency range, in order to reproduce the rigor of launch. These vibration tests need to be conducted in order to ensure that the devised structure meets the expected loads of its future application. However, the structure must not be overtested to avoid any risk of damage. For this, the system's response to the testing loads, i.e., stresses and forces in the structure, must be monitored and predicted live during the test. In order to solve the issues associated with existing methods of live monitoring of the structure's response, this paper investigated the use of artificial neural networks (ANNs) to predict the system's responses during the test. Hence, a framework was developed with different use cases to compare various kinds of artificial neural networks and eventually identify the most promising one. Thus, the conducted research accounts for a novel method for live prediction of stresses, allowing failure to be evaluated for different types of material via yield criteria.

Keywords: mass operator; machine learning; structural stress; artificial neural network; live prediction; vibration test

1. Introduction

In the space industry, the launch evidently dominates structural requirements. Therefore, in order to demonstrate that a structure will survive the launch, it is analyzed using the finite element method (FEM) and tested in vibration test facilities [1]. During a vibration test, accelerations are usually monitored in order to assess the loads that the structure is experiencing. Ideally, load cells are also installed at the interface of the structure to directly monitor the interface loads and compare them against the design loads. This, however, is not always possible because the use of load cells or strain gauges has many technical, operational, and financial drawbacks [2]. Consequently, the input of the vibration test, i.e., the excitation load of the structure under test, is often adjusted based on the measured accelerations rather than on loads or stresses [3].

One specific example concerns the case where loads need to be monitored at the interface of a subsystem that is part of a larger complex system such as the James Webb Space Telescope (JWST) (Figure 1). The JWST is composed of several subsystems, each of which was tested separately before integration on the JWST. Figure 1 illustrates this problem where, particularly on the bottom, are depicted all the different mechanical test campaigns in which the Near-Infrared Spectrograph (NIRSpec) has been involved. One can observe the NIRSpec optical assembly stand-alone test (OA), followed by the integrated science and instrument module test (ISIM) and the optical telescope assembly test (OTE + ISIM = OTIS). The last mechanical test prior to launch has been recently conducted, the observatory test with JWST in folded configuration. Beside these mechanical tests, many more test campaigns have been conducted. This path highlights the path followed only by NIRSpec.

In the later testing phase, it is not possible to monitor the loads at the interface of the NIRSpec using load cells because there is no space for accommodating them [4]. Therefore, alternative approaches must be used.

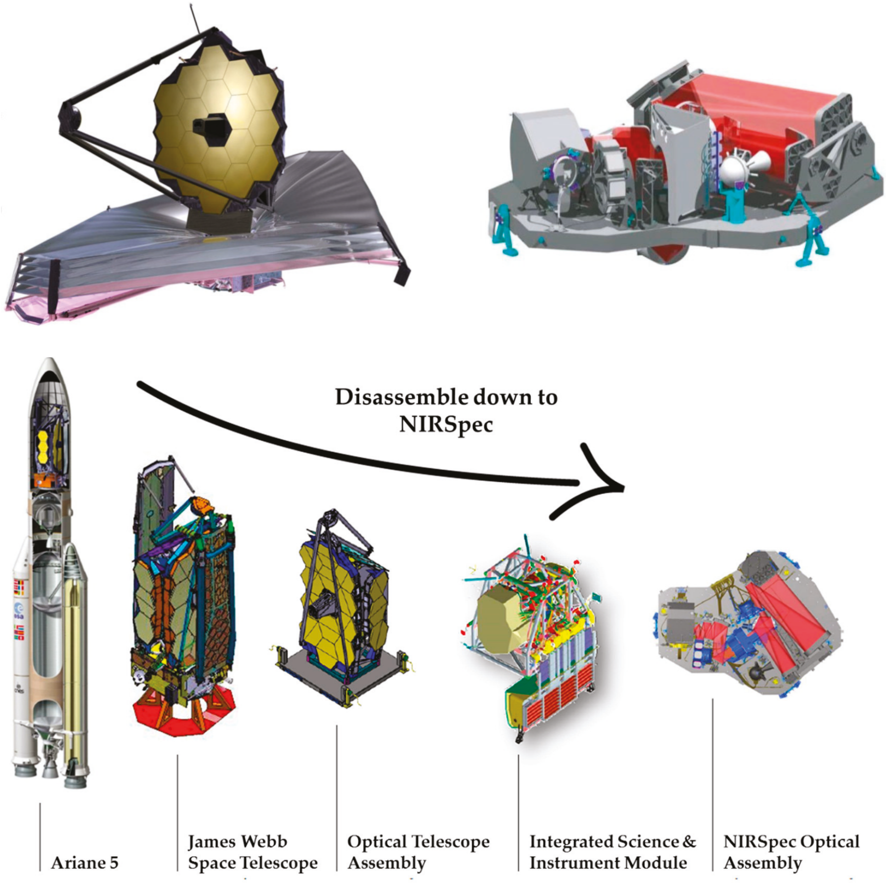

Disassemble down to NIRSpec
Ariane 5
James Webb Space Telescope
Optical Telescope Assembly
Integrated Science & Instrument Module
NIRSpec Optical Assembly

Figure 1. The James Webb Space Telescope (JWST) (top left) in deployed configuration and Near-Infrared Spectrograph (NIRSpec) (top right). On bottom, JWST is disassembled towards NIRSpec [5].

One approach is to use the coil current from the shaker, since the applied load can be correlated with the shaker current. However, this approach can only be used to estimate the load in the excitation direction [6]. Strain gauges could be used to recover strains at interfaces and thus loads. However, they require careful calibration to provide a robust indirect measurement of the interface loads. A force measurement device provides six global interface forces or moments and local load cell forces during vibration testing, allowing the measurement of the local forces in three orthogonal directions [6]. However, such devices are not available in every test facility center. Moreover, they are costly, take space that is not taken into account in the design, and often change the system's response, so they must be accounted for in all test prediction analyses [2,5].

The mass operator is a mathematical tool used to derive loads from measured accelerations [3]. It uses measured accelerations in order to calculate the interface loads or stresses representative of the physical state of a structure. A simple example of a mass operator approach is the sum of weighted accelerations (SWA), which is nothing else but the application of Newton's second law $\mathrm{F} = m\mathrm{a}$, where a is a vector of measured accelerations, m is an equivalent mass matrix, and F is the vector of loads at chosen interfaces. Generally, the mass operator would be created before a vibration test based on finite element analyses' results. The actual computation of the mass matrix can be performed using one of several techniques. With these data, it is then possible during the vibration test to calculate interface loads based on the real-life accelerations, measured by the sensors with no additional hardware [2,3,5]. The authors of [3] provide an extensive review and comparison of mass operators, among them the fitted SWA, the frequency-dependent SWA, and the artificial neural network (ANN).

The fitted SWA is the most straightforward method to calculate mass coefficients. It consists of defining the mass coefficients as design variables of a minimization problem or a curve fitting problem [2,3] where the error E between the response calculated with the finite element method and the response provided by the mass operator is minimized as follows [3]:

$$E = \min\left\{\frac{1}{2}\left|\mathrm{F}_{FEM}(\omega) - \mathrm{F}_{MOP}(\omega)^2\right|\right\}. \tag{1}$$

However, this method works well only over small frequency ranges with few modes. To solve this issue, the authors of [3] considered a frequency-dependent SWA where the frequency range is split into subranges and a fitted SWA is created independently for each subrange. This method is however not well suited to closely spaced modes. In order to generalize the definition of the mass operator, the authors of [3] presented the use of an artificial neural network (ANN) in two different approaches. The ANN can be used to calculate mass coefficients based on input frequencies and accelerations; this is then a generalization of the frequency-dependent SWA. The ANN can also be used to directly provide the force from accelerations and frequency inputs; this is the most general definition of an operator that can convert measured accelerations into quantities of interest such as forces. Both approaches showed great potential for load estimations [3], but the latter approach has shown many drawbacks especially regarding the ability to generalize the mass operator as an ANN, if the tested structure differs from the analyzed one due to uncertainties such as boundary conditions or material properties. Furthermore, a mass operator as an ANN has not been investigated for the estimation of internal structural stresses.

In the last few years, ANNs have shown many successful applications in various domains, from monitoring structural health [7] to predicting tool life [8]. In [9], convolutional ANNs are used to predict vibrations. In a civil structure, vibration-based structural damage can meanwhile be detected using methods based on machine learning [10]. This paper aims to contribute to this expanding field in structural mechanical engineering by expanding the work done in [3] on the use of ANNs. First and foremost, research work was performed on a large-scale structure within an industrial environment. Second, in addition to standard responses such as acceleration response, stresses were successfully predicted. Moreover, several types of neural networks were investigated that could be used to directly convert measured accelerations into structural stresses and hence enable the live prediction of stress during the vibration test. First, the general methods and considered ANNs are presented. Then, a use case is considered in order to test the different ANNs and get a better understanding and confidence about their ability to predict interface loads or stresses in a robust way. Finally, the paper concludes with a discussion on the findings and potential operational use of the proposed approaches.

2. Materials and Methods

In practice, mass operators are built using accelerations and stresses or loads. In this case, the accelerations, loads, and stresses were computed using the finite element method [11,12]. Once the mass operators were built and verified, they were deployed during the test to compute stresses and loads based on measured accelerations. In this paper, only ANNs are considered for creating mass operators and MATLAB 2018b (Mathworks, Natick, MA, USA) was used to create and train the proposed ANN.

A prediction of the structure's response is indeed provided by the finite element analysis (FEA) data. The FE model in this specific case needed to comprise two main aspects. One aspect was the accurate modeling of NIRSpec's ceramic bench, as this is the instrument being designed by AIRBUS and is one of the most sensitive parts. The other aspect was the compliance of the surrounding structure, i.e., the structural elements onto which NIRSpec was mounted. The latter aspect was addressed by conducting a coupled load analysis (CLA), where NIRSpec was considered via a standard FE model and the remaining ones, for instance, the instrument module, the optical telescope, and space craft elements, were represented through stiffness representative super-elements. From this CLA, only the forces and moments acting on NIRSpec were derived. In order to have the full picture, phase information was considered as well in order to depict the dynamical compliance of the overall structure. Next, the interface load input was condensed by only considering frequency steps in the vicinity of peaks in direct response as well as in cross-response. This condensation reduced the input size from roughly 42,000 frequency support points down to 700 (1.7%). This, evidently, reduced the computational efforts on our detailed FE model in terms of stress calculation and post-processing considerably, thereby allowing detailed investigation at mechanically interesting frequency ranges to address the first aspect of our FE approach, namely the detailed stress prediction on our ceramic bench.

However, to use FE models for deriving predictions, one has to assume damping. This highlights the major contributor to potential discrepancies, together with overall system nonlinearities stemming from interface mechanics, secondary structures like harnesses, implemented damper elements, and the like.

This infers that the real-life physical state of the structure, namely the interface forces and stresses, needs to be predicted live during the vibration test based on the actual response of the system. Only then will it be possible to adequately adapt the testing level to protect the structure. Unfortunately, only a limited set of data about the state of the structure is available, such as the measured accelerations at discrete locations on the structure [2]. From these accelerations, the stresses or forces working in the tested structure need to be derived using a dedicated method, such as mass operators or ANNs, which is the subject of this investigation. Any method must meet the following requirements:

- Robustness with regard to natural frequency shifts during testing as compared to the ones computed with FEA.
- Fast deployment during vibration testing in order to react adequately to the resulting responses; the effort of the post-processing model during the test must be small.
- Accuracy for stresses and the interface forces. In this investigation, the von Mises yield criterion was used to monitor the state of the NIRSpec module.
- Fast training, data acquisition, processing, and configuration.
- Robustness with regard to the lack of sensors. As the number of available sensors during the test is restricted, the method must be accurate with a limited number of sensors and, at the same time, potentially inconveniently positioned ones [2,3].

Artificial neural networks (ANNs) mimic the human brain in its mechanisms to transfer data from one neuron to another (see Figure 2). They consist of a connection of different layers where each layer has a defined number of neurons. A neuron is similar to a computing block defined by an activation function, a set of weights and biases, an input, and an output. For more complex problems, a number of hidden layers can be inserted. Data are propagated through the ANN and the output of each layer represents the input of the next layer. The input to an ANN usually comprises the features and the targets. The feature data are used to predict the target data. In the case of mass operators, the features are the accelerations while the targets are the stresses. Such an ANN architecture can be described as a feedforward neural network [13]. If p is considered to be the input to a neuron and b the neuron's bias, then the output of that neuron is $a = f(w \cdot p + b)$, where f represents the neuron activation function and w is a weighting factor. While f is chosen with regard to the problem to be solved, w and b are both parameters that will be calculated based on a learning rule during the training [13]. During training, the network's neurons are first initialized, i.e., a random set of weights and biases is attributed to each neuron and an activation function needs to be assigned to each neuron in the layer. Then, the training data are forward-propagated through the network; each neuron applies its random weights and biases and its activation function to the input and produces an output, which is further propagated until the data reach the output layer. Afterward, an error function E is evaluated, usually the mean squared error (MSE) between the calculated outputs Y_i and the target values T_i:

$$E = MSE = \frac{1}{n}\sum_{i=1}^{n}(T_i - Y_i)^2 \text{ with } n = \text{number of data points.} \tag{2}$$

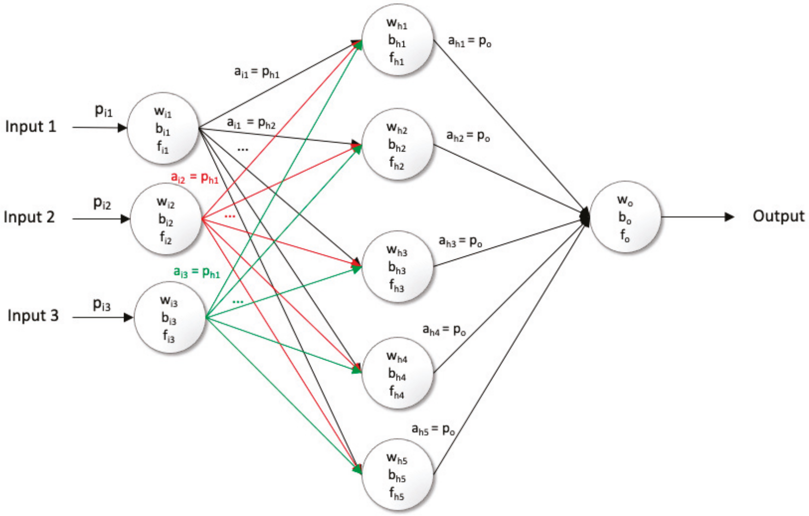


Figure 2. Example of the artificial neural network (ANN) structure with 3 neurons in the input layer, 5 hidden neurons in 1 hidden layer, and 1 output layer and the connectivity between neurons.

Finally, the error is back-propagated through the network in order to identify the neurons that are responsible for the error. The latter are then adapted to minimize the error, specifically their weights and biases are altered, while the connections of the neurons producing a low error are reinforced in this process [14].

The recurrent ANN is capable of exhibiting a dynamic behavior where the output of one layer can also be used as the input for a preceding layer. This makes it then possible for the neural network to create a temporary memory and process sequences of inputs [15]. This is particularly relevant as vibration tests are performed using frequency sweep where, for example, the frequency increases with time.

In this study, four different neural network models are compared to each other:

- A frequency-dependent ANN (see Figure 3a): a feedforward ANN with the frequency values as additional feature data as in [3]. Thus, it is ensured that the data are associated with the corresponding frequency.
- A pretrained ANN: a feedforward ANN trained in two steps, in order to give special attention to the natural frequencies, which represent the most critical frequencies during the vibration test with respect to accelerations and stresses.
- A nonlinear autoregressive exogenous (NARX) model (see Figure 3b): a recurrent ANN to depict the sequence nature of the input data, taking into account the last time step before making a prediction about the next one. For a nonlinear autoregressive exogenous (NARX) model, besides

the external feature sequence u_t , u_{t-1}, u_{t-2}, u_{t-3}, . . ., the targets y_t of the network are also used as features, while a delayed version of them y_{t-1}, y_{t-2}, y_{t-3}, . . . is fed back into a feedforward network, according to [16] by $y_t = f(y_{t-1}, y_{t-2}, y_{t-3}, \ldots, u_t , u_{t-1}, u_{t-2}, u_{t-3}, \ldots)$. While the benefit of such an ANN is its memory of the past values, the disadvantage of the NARX model is that each time step t of the sequence is treated as an independent layer. This can lead to an extremely deep ANN, resulting in an increase in computational time.

- A recurrent ANN with a bidirectional long short-term memory layer (biLSTM): a recurrent ANN with a biLSTM layer to depict the sequence nature of the input data, taking into account both the last as well as the following time step for every prediction. The biLSTM layer is built up by a cell state and three different gates, namely the input, the output, and the forget gate. From this structure, an ANN with an LSTM layer is able to work with a memory. The prefix bi comes from the fact that it is able to use data from prior as well as following time steps. The input gate determines how much of a new value is used as input into the cell, while the forget gate determines how much of the cell state is to be forgotten, and the output gate determines how much of the cell state is used to compute the cell state of the next cell. These elements are combined through several functions as well as matrix operations. More information regarding the mechanisms of biLSTM layers can be found in [17].

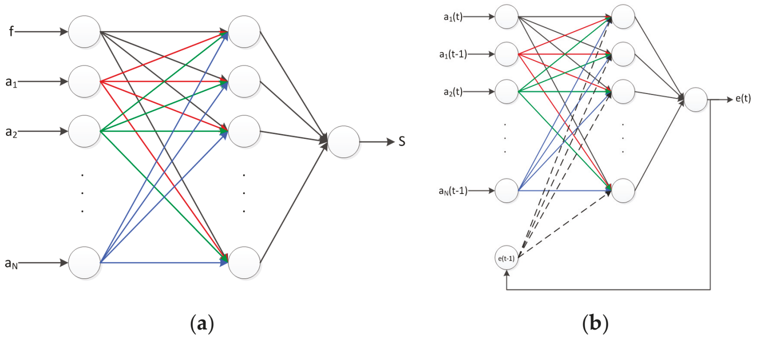


Figure 3. Architecture of the considered ANN: (**a**) frequency-dependent ANN and (**b**) nonlinear autoregressive exogenous (NARX) model.

2.1. Data Generation

The data used to develop the proposed method represent the harmonic response of the system over the frequency range over which the structure will be tested, typically 5 Hz to 100 Hz. In this study, in order to train and evaluate the networks to compare the different ANNs, data had to be generated for the three different scenarios. The training, testing, and validation data were generated by conducting a finite element harmonic analysis to compute the accelerations and stresses or forces at given nodes and elements, respectively, over a determined frequency range (5–200 Hz, step of 2 Hz).

This data set was complemented by another set of data that was generated by conducting a finite element harmonic analysis over the same frequency range, with the same structure but different material properties. The Young's modulus of the JWST's optical bench was decreased by 5% in order to shift the natural frequencies of the structure, and to account for material property uncertainty. The remaining material properties were left unchanged. These artificial data helped the trained models to generalize and make better predictions when the material of the test structure was not identical to the material data considered in the finite element analysis.

The data were then divided into a training set, a testing set, and a validation set to enable the assessment of the training progress and process. In order to improve training, the data at natural frequencies of the structure were included in the training data set, while the remaining frequencies were randomly distributed between the training and the validation data set. Thus, it was ensured that the model learned the connections at the natural frequencies that were the most critical, since the structure experiences the stresses with highest amplitudes. In general, a small random number of frequencies can also be used as a test set to evaluate the model's accuracy. However, in this investigation, the models were assessed on independently generated test data with a changed Young's modulus. In this way, uncertainties, as experienced in reality, were taken into account.

2.2. Data Processing

To improve training and reduce the complexity of the problem to be solved, while increasing accuracy and speeding up the training process, the data of the various observations should be normalized. Every observation was scaled to be in a range from minus one to one. To make usable predictions during the test, the scaling parameters should be stored to denormalize the predictions to real-life figures [13].

2.3. Academic Use Case

For the first scenario, the theoretical case consisted of a very simple structure. It served as a benchmark to determine whether the method would be successful. The structure used for this scenario can be seen in Figure 4. The accelerations of 68 of the 90 nodes of the structure were used to predict the base force of the structure in element 100 (highlighted in Figure 4). The use of this excessive and unrealistic number of sensors (which, in reality, is never the case) enabled the assessment of the overall feasibility of the method. In the case where the method failed to predict the structure's base force, it could be deemed impractical. Furthermore, for this first scenario, the base force and not the stress was to be predicted using the accelerations because its relation to the measurable acceleration is more straightforward.

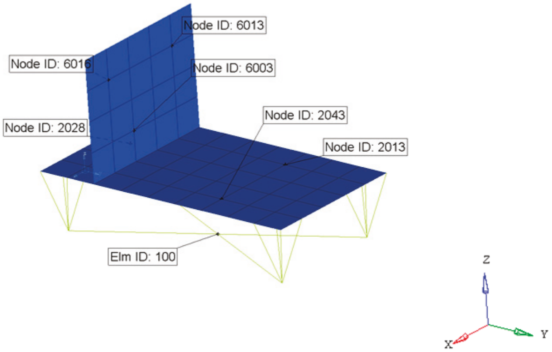


Figure 4. The academic use case with six highlighted sensors and element 100.

The second scenario basically represented a variation of the first scenario, where only six sensors were used to predict the base force as highlighted in Figure 4. This reduced number of sensors reflects reality, where the number of available measuring points is highly restricted. Thus, it provides the possibility to estimate the method's performance in a more realistic case with a limited number of sensors.

2.4. Industrial Use Case

Last but not least, the NIRSpec use case represented an application of the method on a real and complex structure with a reduced number of sensors while predicting the element stress. Consequently, in the case where the models are able to make accurate predictions for those three scenarios, the method can be concluded as useful.

The considered use case scenario concerns an actual structure corresponding to the NIRSpec instrument's optical bench. The optical bench is equipped with ten sensors to predict the stress in one element (see Figure 5). This case makes it possible to evaluate the potential of the method for a real and complex structure with more complex eigenmodes and a limited number of sensors. In this use case, the stress is to be predicted because it represents a good indicator for the structure's physical state and enables the evaluation of the model's performance to predict other metrics than the force, as in [2].

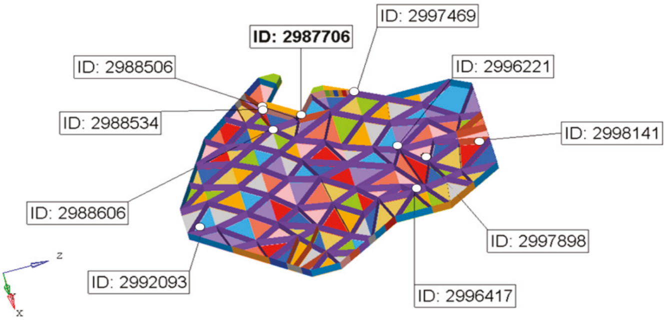


Figure 5. NIRSpec module's optical bench with the ten most stressed elements highlighted, with a perspective from below in order to depict all ribs stiffening the bench.

In order to determine the stresses at the highlighted elements in Figure 5, a FEA was conducted with MSC NASTRAN version 2018.1.0. The structure was discretized by 96,073 nodes and 104,182 elements spanning from one-dimensional elements (i.e., rods and beams) over shell (triangular and quadrangular) to solid elements (tetrahedral, hexahedral, and pentangular). As boundary conditions, the FEA was subjected to forces and moments for each kinematic mount derived from the CLA, where phase information was provided as well. This approach is referred to as the multi-excitation method (MEM). All dynamic analyses were based on modal decomposition, and they are therefore modal frequency response analyses ranging from 5 Hz to 200 Hz. For each of these frequency steps, the von Mises stress was evaluated at the ten selected elements and used to train the ANN. It should be noted that this equivalent stress was used for this paper only. AIRBUS has developed a dedicated equivalent stress suited to predicting ceramic failure.

3. Results

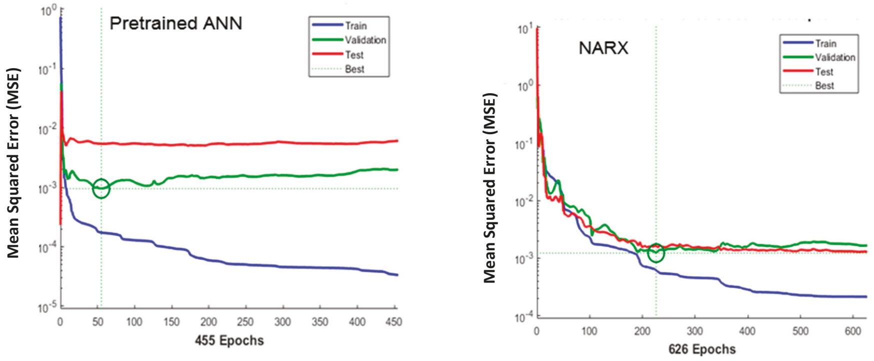
Table 1 summarizes the number of neurons for the different models. The ideal number of neurons was determined in a trial and error way, aiming for the best performance of the MSE while keeping the number of neurons small. The number of delays of the NARX model was determined in the same way. As objective function, the mean squared error (MSE) was used for all the models. While the input differs for each model (see Table 1), the element stress or the base force were used as feature data for all ANN. Furthermore, except for the biLSTM, the Nguyen–Widrow layer initialization function [18] was used to generate the initial weights and biases of the neurons for all ANNs. For the biLSTM, the input weights were initialized with the Glorot/Xavier initializer [19], using an orthogonal initialization for the recurrent weights, while the forget gate bias was initialized with ones and the remaining biases with zeros. All models also share the same activation function, namely the hyperbolic tangent sigmoid, except for the biLSTM, which uses the sigmoid function for the gate, the hyperbolic tangent function for the cell state and hidden state, and the linear activation function for the regression layer. The used training algorithm is also indicated in Table 1.

Table 1. Architecture of the different ANNs.

	Frequency-Dependent ANN	Pretrained ANN	NARX	biLSTM
Inputs	Accelerations Frequency	Pretraining with eigenvectors Accelerations for training	Accelerations as time sequence Feedback	Accelerations as time sequence
Training algorithm	Levenberg–Marquardt backpropagation	Levenberg–Marquardt backpropagation	Scaled conjugate gradient backpropagation	Adam optimizer
Number of nodes for 1st theoretical case	12	12	12 neurons, 3 delays	12
Number of nodes for 2nd theoretical case	19	100	140 neurons, 3 delays	68
Number of nodes for use case	70	5	5 neurons, 1 delay	71

The NARX model was designed in open-loop form, where the input targets were used as feedback features. The model used as many inputs as sensors and had one hidden layer, a defined number of delays, and one output layer per stress (see Table 1). The network with a biLSTM layer consisted of a sequence input layer with as many neurons as inputs, followed by a biLSTM layer. Then, there was one fully connected layer and, lastly, the regression output layer with its linear activation function and as many neurons as outputs.

After the setup of the architecture of the different models, they were trained on the setting as listed in Table 1. Figure 6a,b shows an example of the learning curves for the pretrained ANN and the NARX model for the industrial use case, respectively.



(**a**) (**b**)

Figure 6. Training plots of (**a**) the pretrained ANN and (**b**) the NARX model for the industrial use case.

The blue curves represent the MSE over the training epochs for the training data, the green curves represent the error for the validation error, and the red line represents the error for the test data. The green circle marks the optimal validation performance. The training curves of every model decrease (as clearly shown in Figure 6 for the pretrained ANN and the NARX model), indicating that the models are able to learn the underlying data. The remaining gap between the validation curves and the training curves can be ascribed to the generalization of the data. As the final validation error is not too large, training can be concluded to be successful. After the training, the models were deployed on the test data. The results of their predictions can be seen in the following section.

The evaluation of the training on the theoretical cases shows that the NARX model is extremely sensitive to the resolution of the frequency range. Dividing the frequency range into 600 rather than

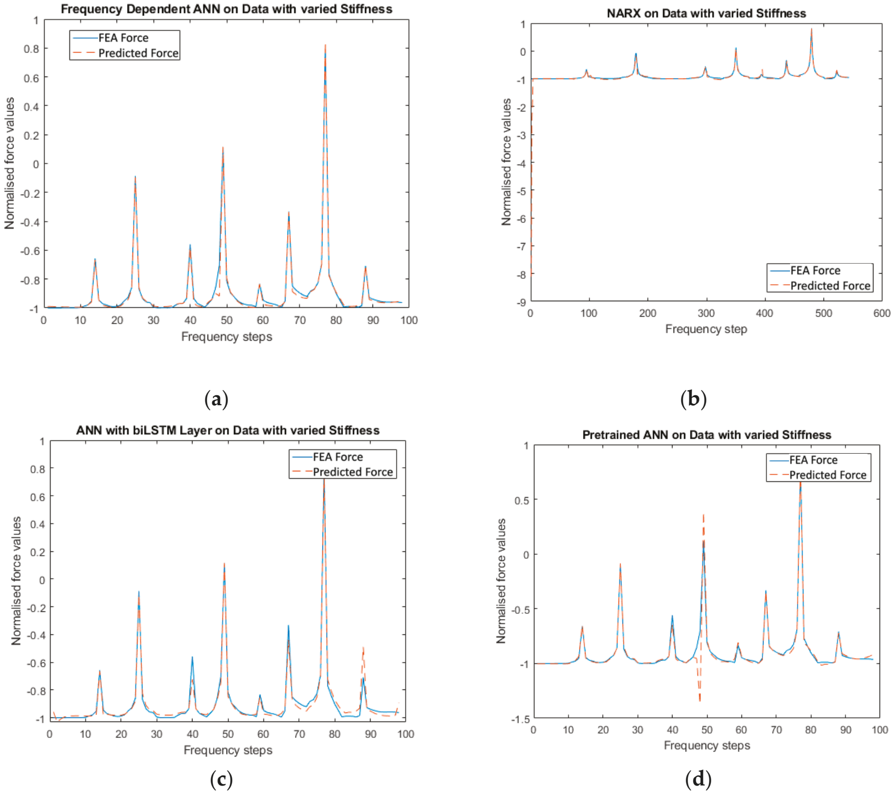
100 steps proves to increase the quality of training tremendously. This does not make a difference for the feedforward and the biLSTM networks, as it only increases computation time.

4. Discussion

In this section, the results of the three different use cases are discussed. Therefore, the different models' predictions of the test data are compared to the FEA and evaluated with a regression analysis.

4.1. Theoretical Case with 68 Sensors

The trained models were deployed to make predictions using the testing feature data. These data were generated by conducting the second FEA and reducing the Young's modulus of the academic structure's material by 5%. As can be seen in Figure 7b, the NARX model makes inaccurate predictions of the first three frequency steps. These steps were used as delays for training. The remaining frequency steps are predicted accurately. The ANN with biLSTM layer was the most delicate to train, and it makes more or less accurate predictions. It wrongly predicts the heights of some peaks, for instance, the peaks at frequency steps 40 and 90, as can be seen in Figure 7b. The frequency-dependent ANN predicts the heights of the peaks correctly (see Figure 7a), whereas the form of the peak at frequency step 50 is poorly predicted. The pretrained ANN (Figure 7d) makes slightly inaccurate predictions about the height and the form of the peak at frequency step 50 as well as the peak at step 90, corresponding to the peaks that are shifted the most in the testing data set compared to the training data set.



(**a**)

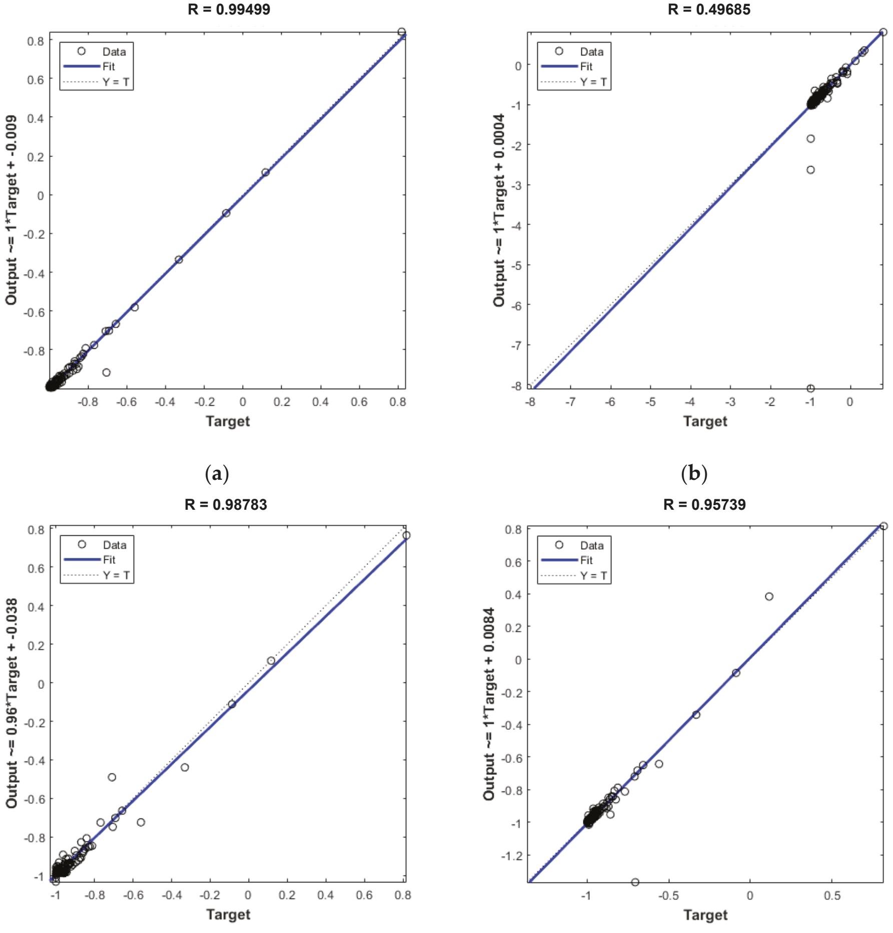
(**b**)

(**c**)

(**d**)

Figure 7. Actual and predicted element force over frequency steps for the academic case with 68 sensors: (**a**) frequency-dependent ANN, (**b**) NARX with more frequency steps, (**c**) ANN with bidirectional long short-term memory layer (biLSTM) layer, and (**d**) pretrained ANN.

This evaluation can be illustrated by a regression analysis. Therefore, the predicted values, called output in Figure 8, are plotted against the calculated base force by FEA, referred to as targets, and a regression line is computed. Figure 8 shows the resulting regression plots, where the black dots represent the data points and the blue line represents the regression line.



(**a**)

(**b**)

(**c**)

(**d**)

Figure 8. Regression plots of the predictions for the academic case with 68 sensors: (**a**) frequency-dependent ANN, (**b**) NARX, (**c**) ANN with biLSTM layer, and (**d**) pretrained ANN.

An overview of the respective regression coefficient of the models, i.e., the slope of the regression line in Figure 8 and the root mean square error (RMSE) between the target and the predicted base force, allows a quantitative comparison of the models. Table 2 summarizes these metrics for the prediction on the test data. The second values (R = 0.9644 and RMSE = 0.0305 N) for the NARX model are the regression coefficient and the RMSE, respectively, for the data without the first three values representing the delays. It can be determined that the pretrained model has the lowest performance, whereas the frequency-dependent ANN makes the most accurate predictions.

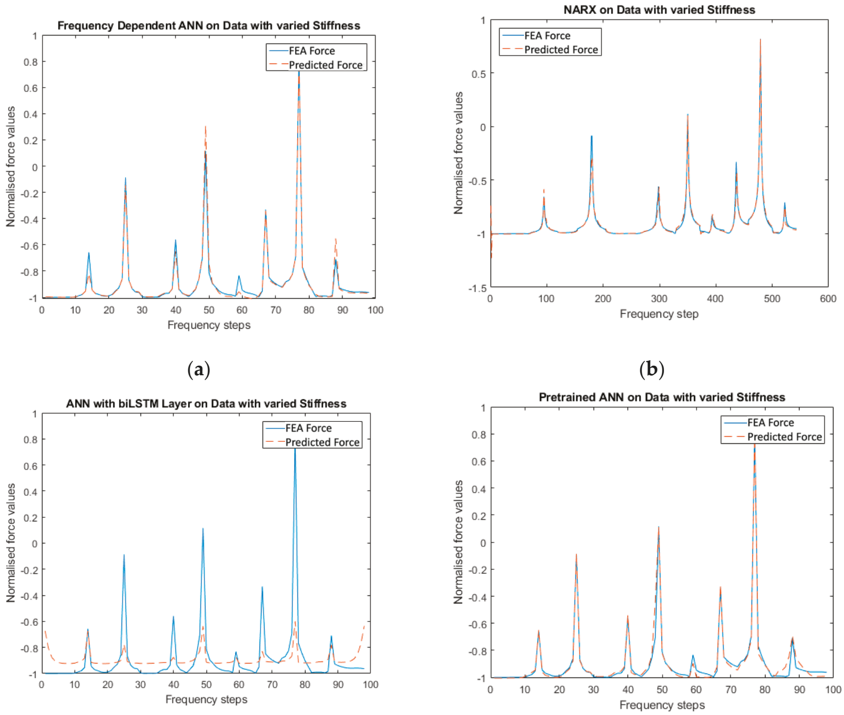
Table 2. Regression coefficients and RMSE for the different models trained on the academic case with 68 sensors.

Model	R	RMSE (N)
Frequency-dependent ANN	0.9950	0.0248
NARX	0.4969/0.9644	0.3164/0.0305
biLSTM	0.9879	0.0353
Pretrained ANN	0.9574	0.2076

The evaluation of the first theoretical case leads to the conclusion that the method has proven to be successful, even though these first predictions were made with an unrealistically large number of sensors.

4.2. Theoretical Case with 6 Sensors

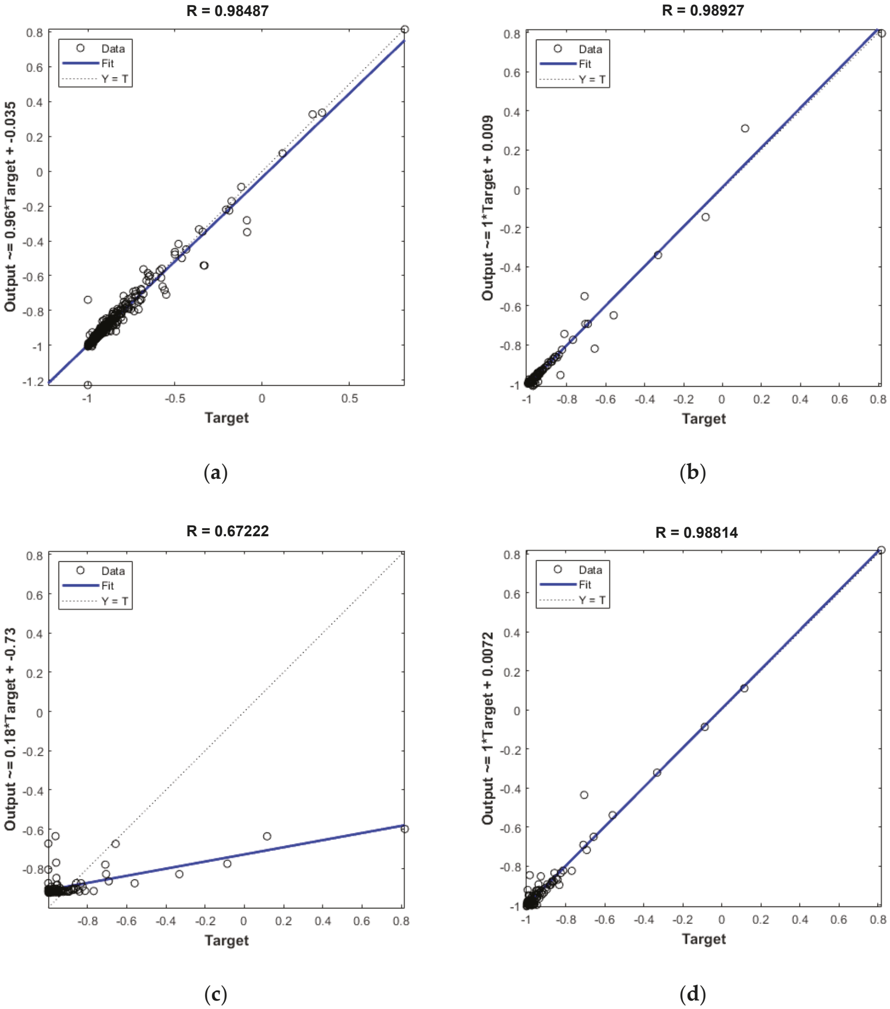
The trained models were deployed to predict the test data with the shifted frequencies, resulting in the predictions seen in Figure 9. While, in this case, the biLSTM was not able to make adequate predictions, as can be seen in Figure 9c, the other models predicted the element force mostly accurately. It can be noted that the NARX model's predictions of the first frequency steps used as delays are not accurate, while the remaining curve is correctly predicted, as in Figure 9b. The pretrained ANN as well as the frequency-dependent ANN make slightly wrong predictions about the height and the form of some of the peaks (see Figure 9a,d).



(**a**) (**b**) (**c**) (**d**)

Figure 9. Finite element analysis (FEA) and predicted element force over frequency steps for the academic case with 6 sensors: (**a**) frequency-dependent ANN, (**b**) NARX with increased number of frequency steps, (**c**) ANN with biLSTM layer, and (**d**) pretrained ANN.

A regression analysis enforces the above observations, as can be seen in Figure 10. The regression coefficients and the RMSE for the predictions on the testing data are summarized in Table 3. While the ANN with biLSTM layer performs worst, resulting from its delicate training, the NARX model makes the most adequate predictions, even despite the delays included in the above calculation. The frequency-dependent ANN and the pretrained ANN perform similarly.



(**a**) (**b**)

(**c**) (**d**)

Figure 10. Regression plots of the predictions for the academic case with 6 sensors: (**a**) frequency-dependent ANN, (**b**) NARX, (**c**) ANN with biLSTM layer, and (**d**) pretrained ANN.

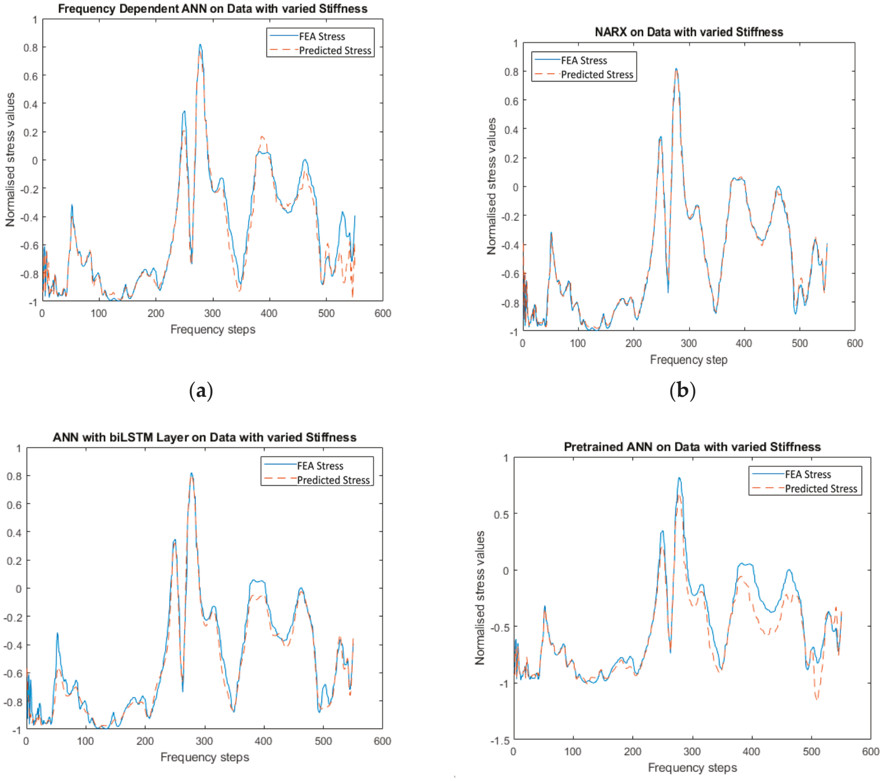
Table 3. Regression coefficients and RMSE for the different models.

Model	R	RMSE (N)
Frequency-dependent ANN	0.9893	0.0366
NARX	0.9849	0.0307
biLSTM	0.6722	0.2037
Pretrained ANN	0.9881	0.0379

This second theoretical case proves that most of the ANNs are also successful in the case, where the number of sensors is restricted, as is often the case in reality.

4.3. NIRSpec Use Case

After the successful training of the ANN models, they were then applied to the NIRSpec use case. The test data with varied stiffness were generated by reducing the Young's modulus of the material of the optical bench plate by 5% of the initial value in the FE model. Figure 11 shows the prediction of the four considered models against the calculated stress with FEA with respect to the frequency steps. The frequency steps divide the considered frequency range (5–200 Hz) into equal steps, the steps and the corresponding normalized stress values both resulting from the FEA.



(**a**)

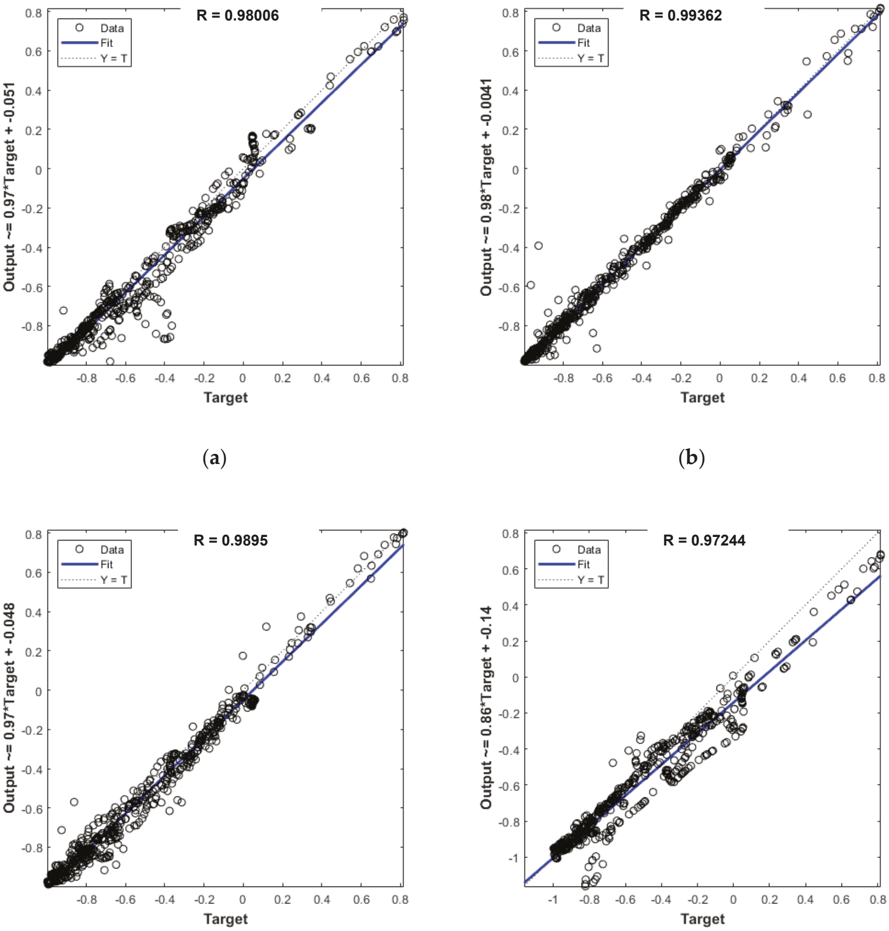
(**b**)

(**c**)

(**d**)

Figure 11. Stress calculated by FEA and predicted stress by ANN over the frequency steps for the NIRSpec use case: (**a**) frequency-dependent ANN, (**b**) NARX, (**c**) ANN with biLSTM layer, and (**d**) pretrained ANN.

As can be seen from Figure 11b, the NARX model predicts the element stress curve without major deviation. It only seems to struggle slightly with the first frequency step, which can be ascribed to the use of that first step as delay in the model's architecture. The frequency-dependent ANN in Figure 11a struggles to predict the peak stress values, for instance, around frequency step 400 and also with the shapes of a few peaks, mainly at the last frequency steps from steps 500 to 600. The pretrained ANN, as seen in Figure 11d, also seems to struggle with the shapes of a few peaks, especially at the end of the frequency range. The fact that both these models struggle at the end of the frequency range can be ascribed to the fact that this represents the mode that was shifted the most by changing the material properties. The NARX model, however, does not face any difficulties with this. Figure 11c shows that the biLSTM also makes more or less accurate predictions, while it was the most delicate to train. However, it also struggles with some peak stress values and completely omits the mode at frequency step 500. Figure 12 shows the resulting regression plots.



(**a**) (**b**)

(**c**) (**d**)

Figure 12. Regression plots of predictions for the NIRSpec use case with 35 sensors: (**a**) frequency-dependent ANN, (**b**) NARX, (**c**) ANN with biLSTM layer, and (**d**) pretrained ANN. The output values and the target values are normalized.

Table 4 makes clear that both recurrent ANNs (NARX and biLSTM) perform the best, the NARX model having the best regression coefficient of 0.9936 and the smallest error of 0.0454 MPa. In contrast, the pretrained ANN makes the largest error of 0.1219 MPa and has the poorest regression coefficient of 0.9724, which is still a high accuracy of prediction.

Table 4. Regression coefficients and RMSE for the different ANN models on the NIRSpec data.

Model	R	RMSE (MPa)
Frequency-dependent ANN	0.9801	0.0862
NARX	0.9936	0.0454
biLSTM	0.9895	0.0656
Pretrained ANN	0.9724	0.1219

5. Conclusions

In this work, four different artificial neural network models were tested for their ability to predict stresses related to the excitation frequency for the launch scenario of the Near-Infrared Spectrograph. In addition, they were tested on a theoretical case with differing numbers of sensors. With correctly trained ANNs, the monitoring of real shaker tests and thus the avoidance of overstressing the test specimens are possible.

The conducted investigation allowed the comparison of all ANN models with respect to the requirements formulated in Section 2. From Tables 2–4 in Section 4, it can be clearly deduced that the NARX model is the most promising one. Figure 7, Figure 9, and Figure 11 illustrate this conclusion. Thus, a trained NARX model could be used during vibration tests and decrease the time of prediction of the given structural parameters, which is crucial for adapting and notching the input load of the shaker in time.

As could also be seen, the recurrent ANN generally performs better than the feedforward ANN, handling the input as concurrent data. The ANN with biLSTM layer is able to make accurate predictions, even though its training is not conducted thoroughly due to the lack of data for a deep ANN. However, if such an ANN is trained with more data and more varied data, it possibly makes the most accurate predictions. In future studies, the potential of this network can be further investigated. For instance, the training data set for this model could be increased by including training data from FEA with several varied Young's moduli or varied damping parameters or by varying other material parameters that have an impact on the natural frequency.

While the NARX model performs the best, its performance is highly dependent on the number of available frequency steps. For example, if the frequency range to be predicted (from 5 to 200 Hz) is poorly resolved and only divided into 100 instead of 600 frequency steps, this situation has a negative effect on the quality of the NARX model's predictions. The other networks are not as sensible to the division of the frequency range. In particular, the ranges of eigenmodes should have a higher resolution by having additional frequency steps. Each time step *t* of the sequence is treated as a single layer, which can lead to an extremely deep ANN. On the one hand, this results in increasing the computational time, but on the other hand it increases the performance of the network. The performance of the NARX model can thus be maximized by training it with as many frequency steps as possible. However, in practice, this can be a hurdle, as the required higher resolution may not be available during the test. Table 5 outlines the qualitative evaluation for the different models in terms of the requirements introduced in the introduction.

Table 5. Evaluation of models for various criteria.

Criterion	Method			
	Frequency-Dependent ANN	NARX	biLSTM	Pretrained ANN
Robustness with regard to frequency shift	~	✓	✓	×
Fast deployment during test	~	~	✓	✓
Accuracy for stresses	✓	✓	✓	✓
Fast training	✓	✓	~	×
Robustness with regard to lack of sensors	✓	✓	×	~

All in all, it can be stated that the conducted research was able to outline a methodology capable of live predicting equivalent stresses of a structure under vibration testing, thereby allowing failure to be evaluated for different types of material via yield criteria.

Author Contributions: L.W. carried out the presented research within her thesis, while R.O., M.S., and K.M.d.P. supported this study as supervisors. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the Structural Analysis Department of AIRBUS Defense & Space GmbH in Immenstaad, Germany, for having supported the research by providing the necessary budget.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wijker, J. *Random Vibrations in Spacecraft Structures Design [Internet]*; Springer: Dordrecht, The Netherlands, 2009; (Gladwell GML, editor. Solid Mechanics and Its Applications; Volume 165); Available online: http://link.springer.com/10.1007/978-90-481-2728-3 (accessed on 14 March 2020).
2. ECSS-E-HB-32-26A. *Spacecraft Mechanical Loads Analysis Handbook*; ECSS: Noordwijk, The Netherlands, 2013; Available online: https://ecss.nl/login/?redirect_to=https%3A%2F%2Fecss.nl%2Fget_attachment.php%3Ffile%3Dhandbooks%2Fecss-e-hb%2FECSS-E-HB-32-26A19February2013.pdf (accessed on 28 November 2020).
3. Olympio, K.R.; Blender, F.; Holz, M.; Kommer, A.; Vetter, R. Comparison of mass operator methods considering test uncertainties. *Adv. Aircr. Spacecr. Sci.* **2018**, *5*, 277–294.
4. Ferruit, P.; Bagnasco, G.; Barho, R.; Birkmann, S.; Böker, T.; De Marchi, G.; Dorner, B.; Ehrenwinkler, R.; Falcolini, M.; Giardino, G.; et al. *The JWST Near-Infrared Spectrograph NIRSpec: Status*; Clampin, M.C., Fazio, G.G., MacEwen, H.A., Oschmann, J.M., Eds.; International Society for Optics and Photonics: Amsterdam, The Netherlands, 2012; p. 84422O. Available online: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.925810 (accessed on 15 March 2020).
5. Jentsch, M.; Schatz, M.; Olympio, R.; Werner, K. An extended mass operator method within James-Webb-Telescope Vibration Testing. In Proceedings of the European Conference on Spacecraft Structures, Materials and Environmental Testing 2018 (ECCSMET), Noordwijk, The Netherlands, 28 May–1 June 2018. paper number 77.
6. Cavro, E.; Capitaine, A. *Proceedings of the 12th European Conference on Spacecraft Structures, Materials & Environmental Testing: 20–23 March 2012, ESTEC, Noordwijk, the Netherlands*; ESA Communications: Noordwijk, The Netherlands, 2012; p. 1.
7. Farrar, C.R.; Worden, K. *Structural Health Monitoring: A Machine Learning Perspective*; Wiley: Hoboken, NJ, USA; Chichester, UK, 2013; p. 631.
8. Oberlé, R.; Schorr, S.; Yi, L.; Glatt, M.; Bähre, D.; Aurich, J.C. A Use Case to Implement Machine Learning for Life Time Prediction of Manufacturing Tools. *Procedia CIRP* **2020**, *93*, 1484–1489. [CrossRef]
9. Liu, J.; Yang, X. Learning to See the Vibration: A Neural Network for Vibration Frequency Prediction. *Sensors* **2018**, *18*, 2530. [CrossRef] [PubMed]
10. Avci, O.; Abdeljaber, O.; Kiranyaz, S.; Hussein, M.; Gabbouj, M.; Inman, D.J. A Review of Vibration-Based Damage Detection in Civil Structures: From Traditional Methods to Machine Learning and Deep Learning Applications. *Mech. Syst. Signal Process.* **2020**, *147*, 107077. [CrossRef]
11. Girard, A.; Roy, N. *Structural Dynamics in Industry*; Wiley: Hoboken, NJ, USA, 2008; p. 421.
12. Craig, R.R.; Benzley, S.E. Structural Dynamics, An Introduction to Computer Methods. *J. Dyn. Syst. Meas. Control* **1982**, *104*, 203. [CrossRef]
13. Hagan, M.T.; Demuth, H.B.; Beale, M.H.; De Jesus, O. *Neural Network Design*, 2nd ed.; Amazon Fulfillment Poland Sp. z o.o: Wrocław, Poland, 2012; p. 800.
14. Ng, A.; Soo, K. *Data Science—Was Ist das Eigentlich?!: Algorithmen des Maschinellen Lernens Verständlich Erklärt*; Springer: Berlin/Heidelberg, Germany, 2018; Available online: http://link.springer.com/10.1007/978-3-662-56776-0 (accessed on 15 March 2020).
15. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; Adaptive computation and machine learning series; MIT Press: Cambridge, MA, USA, 2012; p. 1067.
16. Jain, L.C. (Ed.) *Recurrent Neural Networks: Design and Applications*; The CRC Press international series on computational intelligence; CRC Press: Boca Raton, FL, USA, 2000; p. 392.

17. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
18. Nguyen, D.; Widrow, B. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *1990 IJCNN International Joint Conference on Neural Networks*; IEEE: San Diego, CA, USA, 1990; Volume 3, pp. 21–26. Available online: http://ieeexplore.ieee.org/document/5726777/ (accessed on 23 November 2020).
19. Glorot, X.B.Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; PMLR: Sardinia, Italy, 2010; Volume 9, pp. 249–256. Available online: http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf (accessed on 28 November 2020).

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

PreNNsem: A Heterogeneous Ensemble Learning Framework for Vulnerability Detection in Software

Lu Wang [1,2,3,*], Xin Li [1,3], Ruiheng Wang [1,3], Yang Xin [1,2,3,*], Mingcheng Gao [1,3] and Yulin Chen [2]

[1] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; li_xin@bupt.edu.cn (X.L.); ruiheng@bupt.edu.cn (R.W.); gmc@bupt.edu.cn (M.G.)

[2] Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China; ylchen3@gzu.edu.cn

[3] National Engineering Laboratory for Disaster Backup Recovery, Beijing 100876, China

* Correspondence: wltongxue@bupt.edu.cn (L.W.); yangxin@bupt.edu.cn (Y.X.)

Received: 21 October 2020; Accepted: 6 November 2020; Published: 10 November 2020

Abstract: Automated vulnerability detection is one of the critical issues in the realm of software security. Existing solutions to this problem are mostly based on features that are defined by human experts and directly lead to missed potential vulnerability. Deep learning is an effective method for automating the extraction of vulnerability characteristics. Our paper proposes intelligent and automated vulnerability detection while using deep representation learning and heterogeneous ensemble learning. Firstly, we transform sample data from source code by removing segments that are unrelated to the vulnerability in order to reduce code analysis and improve detection efficiency in our experiments. Secondly, we represent the sample data as real vectors by pre-training on the corpus and maintaining its semantic information. Thirdly, the vectors are fed to a deep learning model to obtain the features of vulnerability. Lastly, we train a heterogeneous ensemble classifier. We analyze the effectiveness and resource consumption of different network models, pre-training methods, classifiers, and vulnerabilities separately in order to evaluate the detection method. We also compare our approach with some well-known vulnerability detection commercial tools and academic methods. The experimental results show that our proposed method provides improvements in false positive rate, false negative rate, precision, recall, and F1 score.

Keywords: cyber security; vulnerability detection; word embedding; deep learning

1. Introduction

Software vulnerabilities are one of the root causes of cybersecurity issues. Despite the improving software quality in academia and industry, new vulnerabilities have been exposed, causing huge losses. A large number of vulnerabilities were proven by Common Vulnerabilities and Exposures [1].

Vulnerability detection is an effective method for discovering software bugs. Overall, vulnerability detection methods can be categorized as static and dynamic methods. High coverage and low false positives are the advantages of static methods and dynamic methods, respectively. Many studies of source-code-based static analysis during the software development stage considered open-source tools [2–4], commercial tools [5–7], and academic research tools [8–10] to reduce dynamic runtime costs. Most of these tools are based on pattern matching. The pattern-based methods require experts to manually define vulnerability features for machine learning or rule matching. In summary, there are two significant drawbacks with the existing solutions: (1) relying on human experts and lacking automation; (2) the high false positive rate and low recall. Both are described below.

The existing solutions rely on human experts to define vulnerability features. It is difficult to guarantee the correctness and comprehensiveness of features because of complexity, even for experts. This is a highly subjective task, because the knowledge and experience of experts influence the results.

It follows that there cannot be a unified standard for manually extracting features. Therefore, we must reduce or eliminate reliance on intense labor from human experts.

The existing solutions produce a high false positive rate and low recall. Most new tools detect all possible vulnerability patterns when matching the rules, regardless of context, structure, or semantics. As such, the detection results have low recall and a high false positive rate. Because of the fixed nature of rule detection, errors occur when detecting the same vulnerability across projects. Although machine learning has been applied to solve the above problems [11,12], the results are still unsatisfactory. These problems suggest that we must achieve a low false positive rate while maintaining a high recall rate.

For the two problems that are mentioned above, the featured engineer should be the core of the solution. Firstly, automated feature extraction will overcome the need for human labor. Secondly, precise vulnerability features will improve the precision of the result. As an automated feature tool, deep learning [13,14] was proposed for vulnerability detection. Applicable deep learning models can automatically and precisely learn various low- and high-level features. However, there are many deep-learning models, and one problem is selecting a model for achieving automation and a lower false positive rate.

In this paper, the proposed framework, which involves pre-training for vector representation, neural networks for automated feature extraction, and ensemble learning for classification (PreNNsem), focuses on improving the feature engineering of vulnerability detection. To validate PreNNsem, we applied different models of pre-training, neural networks, and ensemble learning. Word2vec continuous bag-of-words (CBOW), multiple structural convolutional neural networks (CNNs), and stacking classifiers were found to be the best combination by comparing classification results. In summary, we make four contributions:

1. When compared with our prior work [15], we propose a framework for systematizing feature extraction to automatically detect vulnerabilities based on natural language processing. PreNNsem enables multiple kinds of deep neural networks to extract various kinds of vulnerability features.
2. We transfer the standard language features in an extended corpus to the current model's training and maintain its structural and semantic information through pre-training. We evaluate trainable and non-trainable pre-training methods in terms of detection capability and performance. It is essential to weigh the pros and cons of different pre-training methods for specific vulnerability detection tasks.
3. We compare different neural network models to obtain features of vulnerability, providing a reference for future automated vulnerability detection. Improving the effectiveness of feature extraction, we compose a parallel and sequential architecture neural network.
4. Our proposed method is more effective than the state-of-the-art methods. The experimental results show that PreNNsem is more useful than traditional static analysis tools and state-of-the-art vulnerability detection systems.

The remainder of this paper is structured, as follows: Section 2 reviews related work. Section 3 presents the PreNNsem framework. Section 4 describes our experimental evaluation of PreNNsem and comparison results and Section 5 discusses problems and concludes the paper.

2. Related Work

2.1. Prior Studies Related to Vulnerability Detection

From the degree of automation, previous vulnerability detection methods can be divided into three categories: (i) Manual methods: Many static vulnerability tools, such as Flawfinder [2], RATS [16], and Checkmarx [17], are based on vulnerability patterns, which are defined by human experts. Because pattern matching depends on the rule base, the false positives and/or false negatives are often high. (ii) Semi-automatic methods: Features are manually defined (code-churn, complexity, coverage, dependency, and organizational [18]; code complexity, information flow, functions, and invocations [19]; missing checks [20,21]; and, abstract syntax tree (AST) [22,23]) for traditional machine learning, such as k-nearest neighbor and random forest. MingJian Tang et al. [24] used artificial statistical characteristics to analyze vulnerability trends and dependencies with the Cupra model in multivariate time series. (iii) More automatic methods: Human experts do not need to define features. POSTER [25] presented a method for automatically learning high-level representations of functions. VulDeePecker [13] is a system showing the feasibility of using deep learning to detect vulnerabilities. Venkatraman S et al. [26] proposed a hybrid model by employing similarity mining and deep learning architectures for image analysis. Vasan D et al. [27] analyzed malware images while using a CNN in order to extract features and support vector machine (SVM) for multi-classification.

PreNNsem is an automated approach and an end-to-end vulnerability detection framework. When compared with the manual and the semi-automatic methods, our method abandons subjectivity. Therefore, the features obtained by our method are more persuasive and comprehensive. POSTER extracts features from the level of the function with a coarser granularity. PreNNsem extracts richer features directly from the word level. Compared with VulDeepecker, we have expanded the corpus in the word embedding layer to increase the precision of semantic expression. We used heterogeneous classifiers to improve the stability and accuracy of classification.

2.2. Prior Similar Studies

Pattern-based approach. Z. Li et al. [13] generated vectors from code gadgets using Word2vec, like us. They used Recurrent Neural Network (RNN)-based deep learning and SoftMax for learning classification. Liu S et al. [28] also used RNN for learning high-level representations of abstract syntax trees (ASTs). Duan X et al. [29] extracted semantic features while using code property graph (CPG), obtaining feature matrices by encoding the CPG. Finally, they used attention neural networks for learning classification. Lin G et al. [30] proposed a deep-learning-based framework with the capability of leveraging multiple heterogeneous vulnerability-relevant data sources for effectively learning latent vulnerable programming patterns.

Similarity-based approach. Vinayakumar R et al. [31] used a Siamese network to identify the similarity and deep learning architectures to classify the domain name. Zhao G et al. [32] encoded code control flow and data flow into a semantic matrix. They designed a new deep learning model that measures code functional similarity that is based on this representation. Xiao, Yang et al. [33] used a novel program slicing to extract vulnerability and patch signatures from the vulnerability function and its patched function at the syntactic and semantic levels. Subsequently, a target function was identified as potentially vulnerable if it matched the vulnerability signature but did not match the patch signature. Nair, Aravind et al. [34] examined the effectiveness of graph neural networks for estimating program similarity by analyzing the associated control flow graphs. In [35], they built a graph representation of programs called flow-augmented abstract syntax tree (FA-AST) and applied two different types of graph neural networks (GNNs) on FA-AST to measure the similarity of code pairs.

When compared with any pattern-based approach, the similarity-based approach is sufficient for detecting the same vulnerability in target programs. However, it cannot detect vulnerabilities in some code clones, including deletion, insertion, and rearrangement of statements. PreNNsem is categorized as a pattern-based approach to vulnerability detection. The existing pattern-based approaches have two problems: first, the extracted information's granularity is rough; second, the data set used to learn vulnerability patterns is insufficient. In contrast to the studies reviewed above, PreNNsem has two advantages: first, it directly extracts features from code granularity in order to avoid the loss of information during feature abstraction. Second, by expanding the corpus, it can learn from common programming patterns and improve generalization capabilities.

3. Design of PreNNsem

3.1. Hypothesis

High-level programming languages, like C and JAVA, are designed for humans, and are closed to human expression. They have many similarities with natural language. For example, programming languages are probabilistic in definitions and context-dependent in grammar. Hence, we can borrow concepts from natural language processing (NLP) for vulnerability detection. We consider the concepts of code language and natural language as follows:

- Concepts: A slice of code—sentences, keywords, statements, characters, numbers—words.

  For natural language processing, we encoded each word as a vector and each sentence as a sequence of vectors. Therefore, distributed representations are based on an assumption; words that occur in the same context tend to have similar meanings [36].

  For vulnerability detection, we separated the code segment by tokenization and represented as a sequence of vectors. It has the same form as NLP. Therefore, we made assumptions for vulnerability detection.

Hypothesis 1. *In a programming language, a token's context is its preceding and succeeding tokens. Tokens that occur in the same context tend to have similar semantics.*

Hypothesis 2. *The same types of vulnerabilities have similar semantic characteristics. These characteristics can be learned from the context of vulnerabilities.*

3.2. Overview of PreNNsem

We aimed to automatically detect vulnerability with feature engineering, while using PreNNsem to achieve the goal. Figure 1 shows the process of our proposed framework, in which we take the sliced code as the input, and the output is whether the vulnerability is detected. In this paper, an extended corpus and sample data are transferred from C/C++ source code using security slice [13] and are represented as a sequence of numbers called "vectorize". Subsequently, PreNNsem needs three steps that are related to each other. In this process, the intermediate data serve as the input to the next layer. In the first step, pre-training uses a vectorized extended corpus to generate distributed representation, and the output is a vector of tokens. The embedding layer takes the output as the initialization parameter. In the second step, the sample data pass through the embedding layer. The neural network and the SoftMax layer obtain high-level features. In the third step, supervised learning takes the feature as the input to determine whether the sample is vulnerable.

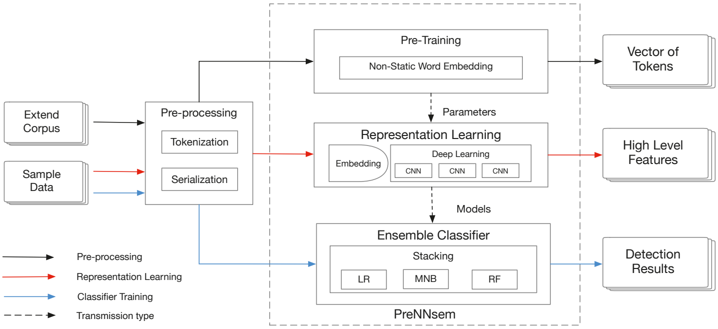


Figure 1. Overview of the proposed PreNNsem (pre-training for vector representation, neural networks for automated feature extraction, and ensemble learning for classification) framework.

3.3. Source Code Pre-Processing

According to code lexical analysis, we remove some semantically irrelevant symbols (e.g., }{() in order to improve efficiency. We divided segment code into words by spaces and symbols (e.g., +−*/=). Deep learning models take vectors (arrays of numbers) as the input. When working with text, we had to develop a strategy to convert strings to numbers before feeding it to the model. Firstly, we indexed each word as a unique number. For example, we assigned 1 to "i", 2 to "for", 4 to "=", 3 to "100", and so on. Subsequently, we encoded the sentence "for i = 100" as a dense vector like [2, 1, 4, 3]. However, different sentences have different lengths. To unify data length for model input, we defined the max fixed-length as 400 according to sample data. There are two cases: if the sentence length is less than 400, zero will be padded; otherwise, the excess will be removed. Note that because pre-training requires a similar representation (Section 3.4) as embedding, extending the corpus only indexes the words in this step.

3.4. Word Embedding Pre-Training

According to Hypothesis 1, the same vulnerability pattern has similar semantics and structure in source code, and code representation is significant for pattern analysis. Word embeddings [37,38] are a type of word representation that allow words with similar meaning to have a similar representation. As such, a similar representation has the same vulnerability pattern. Vulnerability code and non-vulnerability code can be distinguished.

In this section, word embedding is divided into random, static, and non-static [39] embedding, according to the initialization method. Random embedding means all words are randomly initialized and then modified during training. Static embedding means word vectors are pre-trained from distribution representation and kept static and unchanged during training. Non-static embedding means pre-trained vectors from Word2vec are fine-tuned for each task and trained with a deep learning model. We used continuous bag-of-words (CBOW) to obtain densely distributed representation.

How does CBOW work? As shown in Figure 2, CBOW is a three-layer network. Firstly, we convert each word into a one-hot encoding form as the CBOW input. x_{Ck} represents the vectors of surrounding words given a current word x_t, where C is the number of surrounding words and k is the number of vocabulary words. Every x is a matrix with a dimension of $k \times 1$. Secondly, we initialize a weight matrix $W_{k\times d}$ between the input layer and hidden layer. In $W_{k\times d}$, d is a word vector size. In the hidden layer, each x left multiples with W and then adds up to the average as the output h_d of the hidden layer. h_d is a matrix with a dimension of $d \times 1$

$$h_d = \frac{W^T \cdot x_1 + W^T \cdot x_2 + \cdots + W^T \cdot x_C}{C} \tag{1}$$

Next, we initialize a weight matrix $U_{d\times k}$ between the hidden layer and the output layer. In the output layer, h left multiples with U and then adds the $SoftMax$ activation function. y and x have the same dimensions, but each element of y represents each word's corresponding probability distribution.

$$y = SoftMax(U^T \cdot h) \tag{2}$$

The CBOW model is a method of learning. Finally, y is not the last result we want; the intermediate product W is the last word vector. In our proposed method, we define surrounding words windows $C = 10$ and word vector size $d = 200$. According to Figure 2, in CBOW, we want to predict the word of the target location. We use the location's surrounding words as input and then obtain the probability distribution of vocabulary words. Finally, we select the word with the highest probability as the final result. In this process, the weight matrix W is constantly adjusted as the final word vector matrix.

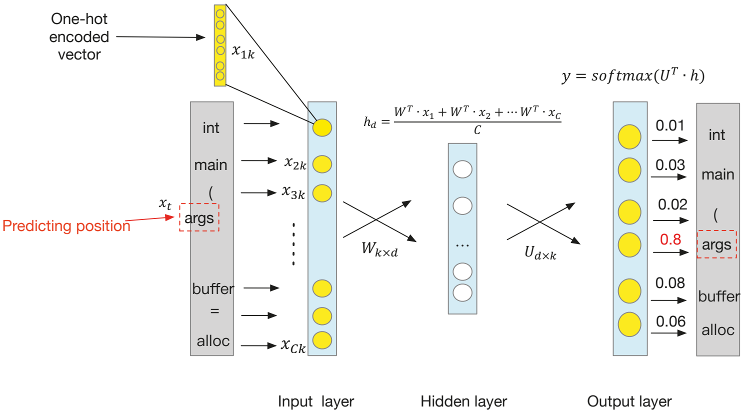


Figure 2. Overview of continuous bag-of-words (CBOW) models.

3.5. Representation Learning

According to Hypothesis 2, common semantic characteristics can be learned from the context of vulnerabilities. Traditionally, the characteristics of manual definition are crucial to machine learning classification. They transform training data and then augment them with additional features to increase the efficacy of machine learning algorithms. However, with deep learning, we can start with raw data, as features will be automatically created by the neural network when it learns.

In this section, we choose CNN and Long Short-Term Memory (LSTM) as the base deep learning model. Figure 3 shows the selected deep learning model used in this study. Firstly, in order to better learn the structure and semantics of the data, we used transfer learning to build the embedding layer for neural networks. Secondly, we sequentially combined three concatenated CNNs and one CNN as a network model. Thirdly, we added a one-dimensional max-pooling layer and a dropout layer for dimension reduction.

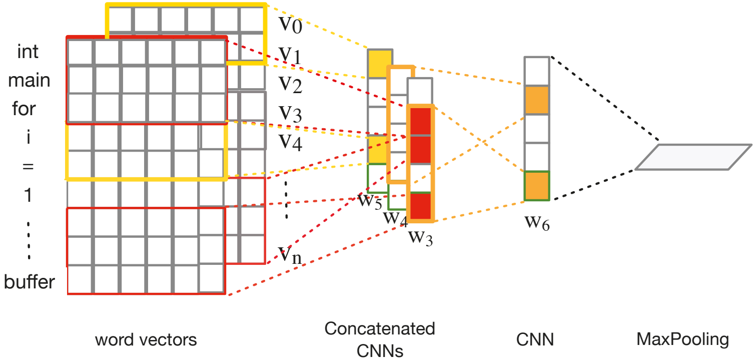


Figure 3. Features of convolutional neural networks (CNNs).

What are the features learned by CNN? As shown in Figure 3, we represent a code segment of length n as:

$$S = [v_0, v_1, ..., v_n] \tag{3}$$

where v_i is the ith word vector in the segment. A filter w_h is used to extract new features combined by the following h words. h is the size of the filter w_h; c_i^h represents the feature generated by combining the ith word and the h words following it.

$$c_i^h = f(w_h v_{i:i+h-1} + b), \tag{4}$$

where f is a non-linear function, b is a bias, and:

$$v_{i:i+h-1} = v_i \oplus v_{i+1} \oplus ... \oplus v_{i+h-1}, \tag{5}$$

where $\oplus$ is the concatenation operator. According to the filter size, there are four different types of filters, including size three, size four, size five, and size six filters. We considered a filter in order to generate a new feature. The larger the filter size, the richer the context of consideration. In our experiment, we applied multiple filters to multiple features. CNN is characterized by parallelism, and each filter is not related to each other, which improves the execution efficiency.

According to Figure 4, LSTM processes one code segment at a time, and the loop allows for information to be passed from one step of the network to the next. This chain-like nature reveals that the recurrent neural networks are intimately related to sequences. They are the genetic architecture of the neural network to use for such data.

In the application of extracting sequence features, RNN can obtain more comprehensive inter-sequence information than CNN. In theory, CNN can only consider consecutive words' characteristics, and RNN can consider the entire sentence. However, in the experiment, the more information stored, the longer the processing time. Even if LSTM has chosen to forget some of the information, there is still the problem of prolonged time consumption for long sequences.

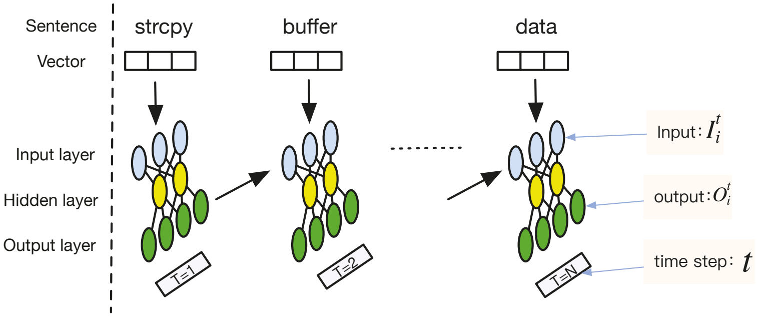


Figure 4. Features of Long Short-Term Memory (LSTM).

3.6. Heterogeneous Ensemble Learning

Recent experimental studies [40] showed that the classifier ensemble may improve the classification performance if we combine multiple diverse classifiers that disagree with each other. Neural network models are nonlinear and have a high variance, which can cause problems when preparing a final model for making predictions. A solution to the high variance of neural networks is to train multiple models and combine their predictions. Ensemble is a standard approach in applied machine learning to ensure that the most stable and best possible prediction is made. We replaced the simple SoftMax classifier with the stacking learning classifier to improve vulnerability classification.

According to [41], heterogeneous ensemble methods have emerged as robust, more reliable, and accurate, intelligent techniques for solving pattern recognition problems. They use different basic classifiers in order to generate several different hypotheses in the feature space and combine them to achieve the most accurate result possible.

How does the stacking framework work? Figure 5 shows the conception of the stacking ensemble. Stacking is used to combine multiple classifiers generated using different learning algorithms $L_1, \cdots, L_N$ on a training dataset S and a testing dataset S', which consist of samples $s_i = (x_i, y_i)$ (x_i: feature vectors, y_i: classifications). Define C as a classifier. Thus,

$$\begin{cases} C_i = L_i(S), & i \in 1, \cdots, N \\ C_{meta} = L_{meta}(S') \end{cases}, \tag{6}$$

where C_i is the base classifiers and C_{meta} is a meta classifier. In the first stage, we choose two base algorithm, $L_1 = LogisticRegression$ and $L_2 = MultinomialNaiveBayesian$. We divide training data into $K = 10$ parts, one of which is the validation subset $S_d, d \in 1, \cdots, K$. We trained C_i on S and evaluated while using 10-fold cross-validation. For the model trained in each step d, we complete predictions on the test set Y'. $\forall i \in 1, \cdots, N$, and $\forall d \in 1, \cdots, K$.

$$\begin{cases} C_i^d = L_i(S - S_d) \\ Y_i^d = C_i^d(S_d) \\ P_i^d = C_i^d(S') \end{cases} \tag{7}$$

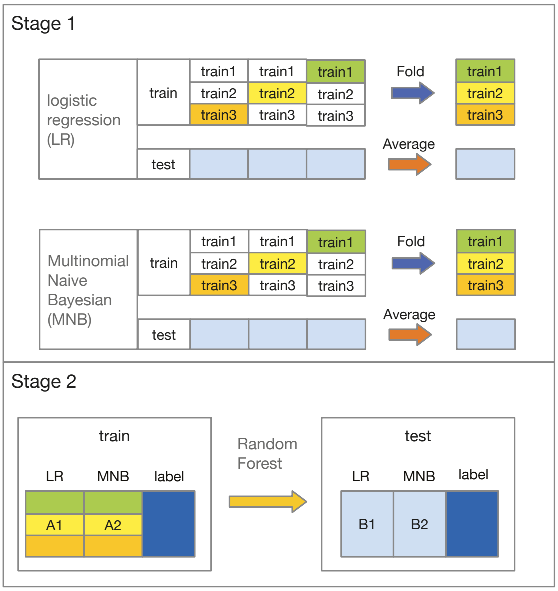


Figure 5. Stacking classifier framework.

Subsequently, each Y_i^d is stacked into a feature A_i. Take the average of all P_i to obtain feature B_i.

$$\begin{cases} A_i = Y_i^1 \uplus Y_i^2 \uplus Y_i^3 \uplus \cdots \uplus Y_i^d \\ B_i = average(P_i^1, P_i^2, P_i^3, \cdots, P_i^d) \end{cases} \tag{8}$$

In the second stage, we concatenate A_i to form a new training data A and concatenate B_i to form new testing data B.

$$\begin{cases} A = A_i \bigoplus A_i \bigoplus A_i \bigoplus \cdots \bigoplus A_i \\ B = B_i \bigoplus B_i \bigoplus B_i \bigoplus \cdots \bigoplus B_i \end{cases} \tag{9}$$

Finally, the meta-classifier is trained on A and predict the result of B.

$$\begin{cases} C_{meta} = L_{meta}(A) \\ Result = C_{meta}(B) \end{cases} \tag{10}$$

3.7. Construct Framework

Now, we build a vulnerability detection framework and propose an implementation. Our proposed framework (PreNNsem) consists of distributed representation, deep learning, and machine learning. We chose an implemented solution, Word2vec CBOW, for distributed representation, multiple structural CNNs for deep learning, and heterogeneous ensemble classifier (stacking) for machine learning.

We tokenize the extended corpus in order to obtain word vectors for similar code representations. Sample data are indexed and sequenced as input to the deep learning model. Word vectors are used as a parameter of the embedding layer. The processed sample data are embedded with neural networks as the input to generate an automatic feature extraction model. Subsequently, features are trained by machine learning and predict whether the samples are vulnerable or not.

4. Experiments and Results

4.1. Evaluation Metrics

Let true positive (TP) denote the number of vulnerable samples detected correctly, false positive (FP) denote the number of normal samples detected incorrectly, false negative (FN) denote the number of vulnerable samples undetected, and true negative (TN) denotes the number of clean samples classified correctly. Running time and memory were considered for testing resource consumption.

We used five metrics to measure vulnerability detection results. The FP rate (FPR) metric measures the ratio of falsely classified normal samples to all normal samples.

$$FalsePositiveRate(FPR) = FP/(FP + TN) \tag{11}$$

False negative rate (FNR) measures the ratio of vulnerable samples classified falsely to all vulnerable samples.

$$FalseNegativeRate(FNR) = FN/(FN + TP) \tag{12}$$

Precision measures the correctness of the detected vulnerabilities.

$$Precision(P) = TP/(TP + FP) \tag{13}$$

Recall represents the ability of a classifier to discover vulnerabilities from all vulnerable samples.

$$Recall(R) = TP/(TP + FN) \tag{14}$$

The $F1$ measure considers both precision and recall.

$$F1 - Measure(F1) = 2 * P * R/(P + R) \tag{15}$$

The low FPR and FNR, and high P, R, and $F1$ metrics indicated the excellent performance in the experimental results. Low resource consumption is also vital.

4.2. Experimental Setup

In terms of collection programs, the Software Assurance Reference Dataset (SARD) [42] serves as the standard dataset to test vulnerability detection tools with software security errors, and the National Vulnerability Database (NVD) [43] contains vulnerabilities in production software. In the SARD, each program case contains one or multiple common weakness enumeration Identifiers (CWE IDs). In the NVD, each vulnerability has a unique common vulnerabilities and exposures identifier (CVE ID) and a CWE ID to identify the vulnerability type. Therefore, we finally collected the programs with CWE IDs that contained vulnerabilities.

We chose two types of vulnerabilities as detection object: buffer overflow (CWE-119) and resource management error (CWE-399). We also collected some other C/C++ programs on NVD as an extended corpus for pre-training. Table 1 summarizes statistics on training data and pre-training data. The datasets were preliminarily processed by [13]. We collected data from the 10,440 programs related to buffer error vulnerabilities and 7285 programs related to resource management error vulnerabilities from the NVD; we also collected 420,627 programs as an extended corpus to improve code representation. The extended dataset focuses on 1591 open-source C/C++ programs from the NVD and 14,000 programs from the SARD. It includes 56,395 vulnerable samples and 364,232 samples that are not vulnerable.

Table 1. Statistics on training data and pre-training data.

Datasets	Samples	Vulnerable	Not Vulnerable	Vocabulary
CWE-119	39,753	10,440	29,313	80,692
CWE-399	21,885	7285	14,600	37,499
Extended dataset	420,627	56,395	364,232	89,642

Regarding training programs vs. target programs, we randomly chose 80% of the programs that were collected as training programs and 20% as target programs. This ratio is applied when dealing with one or both types of vulnerabilities. We also used 10-fold cross-validation over the training set to select the model and used the test set to test the obtained model.

For the deep learning model, we implemented the deep neural network in Python with Keras [44]. We ran experiments on a Google Colaboratory [45] with Nvidia K80, T4, P4, or P100 graphics processing unit (GPU). Genism [46] Word2vec was used to train the word embedding layer. Scikit-learn [47] provides KNeighborsClassifier, RandomForestClassifier, MultinomialNB, and LogisticRegression algorithm as classifiers. Every experiment monitored valid F1 as a condition of early stopping in 10 epochs. Table 2 shows the parameters in the representation learning phase.

Table 2. Tuned parameters for representation learning.

Parameter	Description
Input_dim	The size of sample vocabulary.
Output_dim	The dimensionality of vectors to which the tokens are converted (200).
Sequence_length	The length of each sample (400).
CNN units	There are 4 CNNs with128 filters each, and the sizes of the filters are 3, 4, 5, 6.
Batch_size	The number of samples that are propagated through the network (128).
Loss function	A function to calculate the loss between the predicted value and real value (binary_crossentropy).
Optimizer	The algorithm to optimize the neural network (Adam)
Monitor	The metric to be monitored for early stop (F1) and patience (10).

4.3. Comparison of Different Embedding Methods

We compared CBOW and Skip-gram to verify the effect of the embedding method. Different types of tokens were selected to test the methods. Then, their embedded results were lowered to a two-dimensional diagram, as shown in Figure 6. CBOW performed better. After embedding, semantically similar words are closer to each other in the diagram, which means that word embedding extracts token semantic information in the context code structure. CBOW is more accurate than the information extracted by Skip-gram.

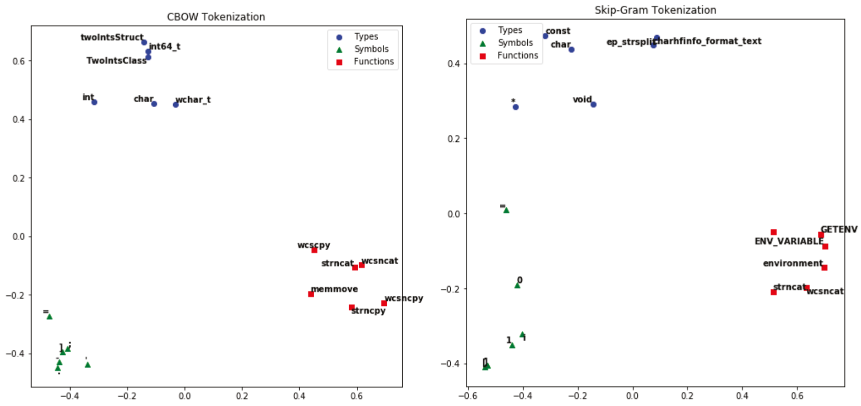


Figure 6. CBOW and Skip-gram tokenization results.

4.4. Comparison of Different Neural Networks

We trained six neural network models on the CWE-119 dataset to evaluate the different neural network models for representation learning. Note that we only indexed and sequenced the dataset instead of vectorizing, so the training dataset is two-dimensional in this section. The models contained: (1) three sequential CNNs with128 filters each; (2) two long short-term memory (LSTM) layers, with a 128-dimensional output; (3) bidirectional long short-term memory (BiLSTM) with a 128-dimensional output; (4) combined CNN (128) and BiLSTM (64 × 2); (5) combined CNN, BiLSTM, and Attention; and, (6) sequentially combine three concatenated convolutional layers and one convolutional layer. To avoid the disappearance of gradients during RNN structural training, in networks that use LSTM, we use sigmoid as the last dense layer activation function, which is different from our previous papers [15]. Table 3 shows the comparison results.

Table 3. Comparison of different representation learning models. CNN, convolutional neural networks. convolutional; BiLSTM, bidirectional long short-term memory; FPR, false positive rate; FNR, false negative rate; P, precision; R, recall; F1, f1-score.

Models	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)
Sequential CNNs	7.2	80.8	48.0	19.2	27.4
Sequential LSTM	12.2	47.5	60.2	52.5	56.1
BiLSTM	10.6	51.5	61.4	48.5	54.2
CNN + BiLSTM	8.0	72.7	54.5	27.3	36.3
CNN + BiLSTM + Attention	8.6	67.5	57.0	32.5	41.4
Concatenated CNNs + CNN	7.4	76.2	52.9	23.8	32.8

Within the margin of error, sequential CNNs and concatenated CNNs achieved the best FPR result. Sequential LSTM has balanced performance and achieved the best results in FNR, recall, and F1. It also has excellent precision. Of the CNNs, concatenated CNNs perform better. Therefore, we tested the embedding layer on sequential LSTM and concatenated CNNs.

4.5. Combination of Different Embedding Methods and Different Neural Networks

According to [39], we divided the pre-training into random, static, and non-static initialization, and then defined the vectors' dimension as 200. Random initialization means that all words are randomly initialized and then modified during training. Static initialization means that all words are pre-trained from Word2vec to generate vectors and non-trainable in work. Non-static initialization means that pre-trained vectors are fine-tuned for each work. We used the CWE-119 dataset and tested different pre-training methods on the sequential LSTM and concatenated CNN models. For training the Word2vec embedding layer, we used CWE-119 as the corpus and SySeVR [48] data as the extended corpus. In this section, we count the memory and training time of the models to compare their resource consumption. Table 4 shows the comparison results.

Table 4. Comparison of different embedding methods on different corpora. Memory, memory consumption; Time, time consumption.

Methods	Corpus	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)	Memory (MB)	Time (s/epoch)
CNN-random	-	3.4	12.3	90.1	87.7	88.9	1675.1	15.0
CNN-static	CWE-119	3.9	16.6	88.4	83.4	85.8	1671.0	9.9
CNN-non-static	CWE-119	2.5	10.7	92.6	89.3	90.9	1694.1	14.9
CNN-static	SySeVR	2.5	13.6	92.4	86.4	89.3	1670.9	10.1
CNN-non-static	SySeVR	2.6	9.7	92.3	90.2	91.2	1673.5	15.2
LSTM-random	-	2.1	12.9	93.5	87.1	90.2	1377.2	200.5
LSTM-static	CWE-119	2.3	13.4	92.8	86.6	89.6	1365.6	183.4
LSTM-non-static	CWE-119	1.9	9.8	94.1	90.2	92.1	1366.8	201.9
LSTM-static	SySeVR	1.6	10.5	95.1	89.5	92.2	1370.4	183.6
LSTM-non-static	SySeVR	2.3	10.7	93.0	89.3	91.1	1373.5	197.1

As shown in Table 4, CNN excelled in terms of FNR and recall, and LSTM excelled for FPR and precision. However, the time consumption of LSTM was 18 times that of CNN. For both, we obtained the following conclusions. According to the corpus, the extended corpus has better metrics because the more words we trained, the more appropriate the obtained vector. According to the false rate (FPR + FNR), P, R, and F1, we found that trainable embedding is better than static embedding because the fine-tuning can be adjusted to each work. The memory of training is almost the same, because the input sample data and the embedding size were the same. Less time was required for static embedding because the increase in trainable parameters leads to increased training time.

In conclusion, when considering the results and efficiency, we chose non-static CNN with extending the corpus as our final deep learning model.

4.6. Comparison of Different Classification Algorithms

Through Section 4.4, we observed that concatenated CNNs are the appropriate deep learning model to extract features. In Table 5, we directly use traditional machine learning after the word-embedding layer. In order to improve the classification results, we chose a different ensemble learning model [49] to substitute the simple activation sigmoid after CNNs. We chose boosting and bagging as our homogeneous ensemble model, including gradient boosting decision tree (GBDT) and random forest (RF). We used stacking for generating ensembles of heterogeneous classifiers, logistic regression (LR) and MultinomialNB (NB) as the base classifiers, and RF as the final classifier. For comparison with ensemble classifiers, we also chose traditional classifiers, including KNeighbors (KN), NB, and LR. Finally, Table 5 shows the comparison results.

Table 5. Comparison of different classification algorithms. ML, machine learning; KN, KNeighbors; LR, logistic regression; NB, MultinomialNB; GBDT, gradient boosting decision tree; RF, random forest.

Methods	Base Classifier	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)	Time (s)
Traditional ML	KN	6.5	22.3	80.6	77.7	79.1	0.7
Traditional ML	LR	5.6	27.1	81.9	72.9	77.1	0.7
CNNs + Traditional ML	KN	3.3	6.6	90.7	93.4	93.0	0.6
CNNs + Traditional ML	NB	5.6	6.1	85.4	93.9	89.5	0.1
CNNs + Traditional ML	LR	2.1	7.9	93.9	92.1	92.9	0.6
CNNs + Boosting	GBDT	1.6	9.1	95.5	90.7	93.0	48.8
CNNs + Bagging	RF	1.6	8.8	95.8	91.2	93.1	10.7
CNNs + Stacking	LR, NB, RF	1.5	8.0	95.4	91.9	93.6	13.7

Table 5 shows that the first two lines did not use representation learning to extract features, and the classification effect was poor. Machine learning with CNNs performed better than traditional machine learning. We concluded that word embedding can only extract the granular features of words. CNNs can obtain the features of code structure, not only word semantics. Therefore, multiple granularity features help to improve the performance of the classifier.

The results of the last three lines (CNN + Ensemble) were generally better than those of lines three to five. Although CNN + NB produced the best recall results (93.9%), its precision was worse, at 85.4%, resulting in an F1 score of only 89.5%, which represents comprehensive performance. Low precision leads to spending more effort and time on the wrong detection results. Therefore, ensemble learning can further improve vulnerability detection. Of the three ensemble learnings, the stacking that was used in this article yielded the best results because it combines multiple diverse algorithms to generate several different hypotheses in the feature space and achieves the most accurate result possible. Though time consumption is higher compared to traditional machine learning methods, we emphasize the detection results for vulnerability detection tasks. Therefore, the increased time consumption is within an acceptable range.

Above all, we selected the most appropriate implementation of PreNNsem through our experiments; it consists of non-static pre-training with an extended corpus, concatenated CNNs representation learning, and stacking classifier.

4.7. Ability to Detect Different Vulnerabilities

As shown in Table 6, the proposed method was applied to the six datasets. We tested our model on the buffer overflow CWE-119 dataset and resource management error CWE-339 dataset in order to evaluate our method's detection ability for different types of vulnerabilities. To validate our approach's generalization capabilities, we selected three different types of vulnerability datasets: Array Usage, API Function Call, and Arithmetic Expression. Each type of dataset contains multiple CWE vulnerabilities. Array Usage (87 CWE IDs) accommodates the vulnerabilities related to arrays (e.g., improper use of array element access, array address arithmetic, and address transfer as a function parameter). API Function Call (106 CWE IDs) accommodates the vulnerabilities related to library/API function calls. Arithmetic Expression (45 CWE IDs) contains the vulnerabilities that are related to improper arithmetic expressions (e.g., integer overflow). Finally, we combined the three to form a hybrid vulnerability dataset, Hybrid Vulnerabilities.

Table 6. Comparison of different classification algorithms.

Vulnerability	Vulnerability Dataset	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)
Specific CWE ID	Buffer Overflow	1.5	8.0	95.4	91.9	93.6
	Resource Management Error	0.3	2.2	99.5	97.7	98.6
Multiple CWE IDs	Array Usage	2.6	7.9	92.3	92.1	92.2
	API Function Call	2.1	9.5	92.4	90.5	91.5
	Arithmetic Expression	1.4	5.2	92.7	94.8	93.8
	Hybrid Vulnerabilities	1.5	6.3	94.4	93.7	94.1

According to the results, we found that the method for detecting specific vulnerabilities performs well. Resource management error has the best result, F1 Score, at 98.6%. Our approach also performs well in detecting the same type of vulnerability. API Function Call has the lowest F1 score, but the result was still no less than 91.5%. The method performed better on hybrid vulnerability datasets than on the same vulnerability datasets, because having more data can improve the model's indicators. In summary, our approach performs well on a variety of data sets.

4.8. Comparative Analysis

We compared our best experimental results with those of state-of-the-art methods in order to verify the performance of the proposed method. We chose open-source static analysis tool Flawfinder [2], commercial static analysis tool Checkmarx [17], vulnerable code clone detection tool VUDDY [50], and academic deep learning methods VulDeePecker [13], DeepSim [32], and VulSniper [29]. Our three reasons for selecting these were: (1) these tools represent the state-of-the-art static analyses for vulnerability detection; (2) they directly operate on the source code; and, (3) they were available to us. Flawfinder and Checkmarx represent manual methods based on static analysis. VUDDY is suitable for detecting vulnerabilities incurred by code cloning. VulDeePecker, DeepSim, and Vulsnipper use deep learning to analyze source code. All of the results in Table 7 are based on the CWE-119 dataset. The results of Checkmarx and VulDeePecker were obtained from [13]. The results of DeepSim and VulSniper were obtained from [29].

Table 7. Comparison of experimental results obtained using the proposed method and those using state-of-the-art methods.

Method	FPR (%)	FNR (%)	P (%)	R (%)	F1 (%)
Flawfinder	46.3	69.0	23.7	40.5	29.9
Checkmarx	43.1	41.1	39.6	58.9	47.3
VUDDY	3.5	91.3	47.0	8.7	14.7
VulDeePecker	2.9	18.0	91.7	82.0	86.6
DeepSim	16.1	41.6	71.6	58.4	64.4
VulSniper	6.42	26.2	88.7	73.8	80.6
PreNNsem	1.5	8.0	95.4	91.9	93.6

Our method outperformed the state-of-the-art methods. Because these traditional tools depend on the rule base, they incurred high FR (FPR and FNR) and lower precision, recall, and F1. VulDeePecker was found to be better than the other tools, with a precision of 91.7%. However, VulDeePecker's recall rate was low, only 82.0%, because it does not expand the corpus during the word embedding phase. DeepSim and VulSnipper extract features from the intermediate code, which loses some of the information. Accordingly, both precision and recall do not work well. Our method automatically extracts vulnerability features directly from the slice source code and does not rely on the rule library. In the word embedding phase, we expand the corpus to obtain richer semantics. Therefore, we improved vulnerability detection capabilities. When compared to VulDeePecker, we improved FPR by 1.4%, FNR by 10%, pPrecision by 3.7%, recall by 9.9%, and F1 by 7%.

5. Conclusions

In this paper, according to existing detection methods, we analyzed vulnerability detection's core problem, which is the lack of proper feature extraction. Firstly, we researched vulnerability detection methods related to deep learning. We then presented the PreNNsem framework to detect vulnerabilities by analyzing source code. We drew some insights that were based on the collected dataset, including explanations for word embedding, deep learning model, and classifier comparisons in vulnerability detection. We used six different vulnerability datasets to prove our method's generalization ability. Finally, we compared the results that were obtained with our method with those of the state-of-art tools and academic methods to validate the improvement in vulnerability detection.

In terms of practicality, our summary is as follows: (i) our method performs well on various mixed vulnerability data sets. Our method can detect various vulnerabilities. (ii) Because we analyze the source code from the perspective of analyzing text, other high-level language source code vulnerabilities can also use our framework. (iii) Each part of PreNNsem also supports other methods, which proves the scalability of the framework.

However, our method has several limitations: (i) our method only focuses on the source program, and our framework can not be applied in executable programs. (ii) Our approach relies on VulDeePecker's [13] code snipping, which will be proposed and integrated into our future framework. (iii) Although we chose several deep learning models, we need to evaluate other models. (iv) The sample length is padded if it is shorter than the fixed length and cut off if it is longer; future works need to investigate how to handle vectors' varying lengths.

Author Contributions: Conceptualization, L.W. and X.L.; methodology, L.W. and X.L.; software, L.W.; validation, L.W.; formal analysis, L.W.; investigation, L.W. and R.W.; resources, L.W.; data curation, L.W.; writing—original draft preparation, L.W.; writing—review and editing, L.W.; visualization, L.W.; supervision, L.W.; project administration, L.W.; funding acquisition Y.X., Y.C., and M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Major Scientific and Technological Special Project of Guizhou Province (20183001), the Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2018BDKFJJ021), the Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2017BDKFJJ015), and the National statistical scientific research project of China (2018LY61, 2019LY82).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. CVE. Available online: http://cve.mitre.org (accessed on 1 March 2020).
2. Flawfinder. Available online: https://dwheeler.com/flawfinder/ (accessed on 1 March 2020).
3. RIPS. Available online: http://rips-scanner.sourceforge.net (accessed on 1 March 2020).
4. Cppcheck. Available online: https://sourceforge.net/projects/cppcheck (accessed on 1 March 2020).
5. Coverity. Available online: http://www.coverity.com/index.html (accessed on 1 March 2020).
6. Fortify SCA. Available online: http://www.fortify.com/ (accessed on 1 March 2020).
7. Ounec5.0. Available online: http://www.ouncelabs.com/ (accessed on 1 March 2020).
8. Cobra. Available online: https://github.com/WhaleShark-Team/cobra (accessed on 1 March 2020).
9. mygcc. Available online: http://mygcc.free.fr (accessed on 1 March 2020).
10. Uno. Available online: http://spinroot.com/uno/ (accessed on 1 March 2020).
11. Yamaguchi, F.; Maier, A.; Gascon, H.; Rieck, K. Automatic inference of search patterns for taint- style vulnerabilities. In Proceedings of the 36th IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 797–812.
12. Pang, Y.; Xue, X.; Namin, A.S. Predicting vulnerable software components through N-gram analysis and statistical feature selection. In Proceedings of the 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 9–11 December 2015; pp. 543–548.
13. Li, Z.; Zou, D.; Xu, S.; Ou, X.; Jin, H.; Wang, S.; Deng, Z.; Zhong, Y. Vuldeepecker: A deep learning-based system for vulnerability detection. In Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 18–21 February 2018.
14. Zhou, Y.; Liu, S.; Siow, J.; Du, X.; Liu, Y. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
15. Li, X.; Wang, L.; Xin, Y.; Yang, Y.; Chen, Y. Automated Vulnerability Detection in Source Code Using Minimum Intermediate Representation Learning. *Appl. Sci.* **2020**, *10*, 1692. [CrossRef]
16. RATS. Available online: https://code.google.com/archive/p/rough-auditing-tool-for-security/ (accessed on 5 March 2020).
17. Checkmarx. Available online: https://www.checkmarx.com (accessed on 5 March 2020).
18. Zimmermann, T.; Nagappan, N.; Williams, L. Searching for a needle in a haystack: Predict- ing security vulnerabilities for windows vista. In Proceedings of the 3rd International Conference on Software Testing, Verification and Validation, Paris, France, 6–10 April 2010; pp. 421–428.
19. Younis, A.; Malaiya, Y.; Anderson, C.; Ray, I. To fear or not to fear that is the question: Code characteristics of a vulnerable function with an existing exploit. In Proceedings of the CODASPY'16: 6th ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9–11 March 2016; ACM: New York, NY, USA, 2016; pp. 97–104.

20. Yamaguchi, F.; Wressnegger, C.; Gascon, H.; Rieck, K. Chucky: Exposing missing checks in source code for vulnerability discovery. In Proceedings of the CCS'13: 20th ACM SIGSAC Conference on Computer & Communications Security, Berlin Germany, 4–8 November 2013; ACM: New York, NY, USA, 2013; pp. 499–510.
21. Thummalapenta, S.; Xie, T. Alattin: Mining alternative patterns for detecting neglected conditions. In Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand, 16–20 November 2009; pp. 283–294.
22. Yamaguchi, F.; Lindner, F.; Rieck, K. Vulnerability extrapolation: Assisted discovery of vulnerabilities using machine learning. In Proceedings of the 5th USENIX Workshop on Offensive Technologies, San Francisco, CA, USA, 8 August 2011; USENIX Association: Berkeley, CA, USA, 2011.
23. Yamaguchi, F.; Lottmann, M.; Rieck, K. Generalized vulnerability extrapolation using abstract syntax trees. In Proceedings of the ACSAC'12: 28th Annual Computer Security Applications Conference, Orlando, FL, USA, 3–7 December 2012; ACM: New York, NY, USA, 2012; pp. 359–368.
24. Tang, M.; Alazab, M.; Luo, Y. Big Data for Cybersecurity: Vulnerability Disclosure Trends and Dependencies. *IEEE Trans. Big Data* **2017**, *5*, 317–329. [CrossRef]
25. Lin, G.; Zhang, J.; Luo, W.; Pan, L.; Xiang, Y. POSTER: Vulnerability discovery with function representation learning from unlabeled projects. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 2539–2541.
26. Venkatraman, S.; Alazab, M.; Vinayakumar, R. A hybrid deep learning image-based analysis for effective malware detection. *Inf. Secur. Tech. Rep.* **2019**, *47*, 377–389. [CrossRef]
27. Vasan, D.; Alazab, M.; Wassan, S.; Safaei, B.; Zheng, Q. Image-Based malware classification using ensemble of CNN architectures (IMCEC). *Comput. Secur.* **2020**, *92*, 101748. [CrossRef]
28. Liu, S.; Lin, G.; Han, Q.L.; Wen, S.; Zhang, J.; Xiang, Y. DeepBalance: Deep-Learning and Fuzzy Oversampling for Vulnerability Detection. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 1329–1343. [CrossRef]
29. Duan, X.; Wu, J.; Ji, S.; Rui, Z.; Luo, T.; Yang, M.; Wu, Y. VulSniper: Focus Your Attention to Shoot Fine-Grained Vulnerabilities. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI-19, Macao, China, 10–16 August 2019.
30. Lin, G.; Zhang, J.; Luo, W.; Pan, L.; De Vel, O.; Montague, P.; Xiang, Y. Software Vulnerability Discovery via Learning Multi-domain Knowledge Bases. *IEEE Trans. Dependable Secur. Comput.* **2019**. [CrossRef]
31. Vinayakumar, R.; Alazab, M.; Srinivasan, S.; Pham, Q.V.; Padannayil, S.K.; Simran, K. A Visualized Botnet Detection System based Deep Learning for the Internet of Things Networks of Smart Cities. *IEEE Trans. Ind. Appl.* **2020**, 56, 4436–4456. [CrossRef]
32. Zhao, G.; Huang, J. DeepSim: deep learning code functional similarity. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Lake Buena Vista, FL, USA, 4–9 November 2018; ACM: New York, NY, USA, 2018.
33. Xiao, Y.; Chen, B.; Yu, C.; Xu, Z.; Yuan, Z.; Li, F.; Liu, B.; Liu, Y.; Huo, W.; Zou, W.; et al. MVP: Detecting Vulnerabilities using Patch-Enhanced Vulnerability Signatures. In Proceedings of the 29th USENIX Security Symposium, Boston, MA, USA, 12–14 August 2020.
34. Nair, A.; Roy, A.; Meinke, K. funcGNN: A Graph Neural Network Approach to Program Similarity. In Proceedings of the ESEM '20: 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Bari, Italy, 5–7 October 2020. [CrossRef]
35. Wang, W.; Li, G.; Ma, B.; Xia, X.; Jin, Z. Detecting Code Clones with Graph Neural Networkand FlowAugmented Abstract Syntax Tree. In Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, ON, Canada, 18–21 February 2020.
36. Pantel, P. Inducing ontological co-occurrence vectors. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2005; pp. 125–132.
37. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
38. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Stateline, NV, USA, 5–10 December 2013; pp. 3111–3119.
39. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.

40. Salunkhe, U.R.; Mali, S.N. Classifier Ensemble Design for Imbalanced Data Classification: A Hybrid Approach. *Procedia Comput. Sci.* **2016**, *85*, 725–732. [CrossRef]
41. Tewari, S.; Dwivedi, U.D. A comparative study of heterogeneous ensemble methods for the identification of geological lithofacies. *J. Petrol. Explor. Prod. Technol.* **2020**, *10*, 1849–1868. [CrossRef]
42. SARD. Available online: https://samate.nist.gov/index.php/SARD.html (accessed on 20 March 2020).
43. NVD. Available online: https://nvd.nist.gov/ (accessed on 20 March 2020).
44. Keras. Available online: https://github.com/fchollet/keras (accessed on 20 March 2020).
45. GoogleColaboratory. Available online: https://colab.research.google.com/notebooks/intro.ipynb (accessed on 20 March 2020).
46. Gensim. Available online: https://radimrehurek.com/gensim/ (accessed on 20 March 2020).
47. Scikit-learn. Available online: https://scikit-learn.org/stable/index.html (accessed on 20 March 2020).
48. SySeVR. Available online: https://github.com/SySeVR/SySeVR (accessed on 20 March 2020).
49. Fang, Y.; Liu, Y.; Huang, C.; Liu, L. FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. *PLoS ONE* **2020**, *15*, e0228439. [CrossRef] [PubMed]
50. VUDDY. Available online: https://github.com/squizz617/vuddy (accessed on 20 March 2020).

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

Article

Using BiLSTM Networks for Context-Aware Deep Sensitivity Labelling on Conversational Data

Antreas Pogiatzis * and Georgios Samakovitis *

School of Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, UK

* Correspondence: a.pogiatzis@greenwich.ac.uk (A.P.); g.samakovitis@greenwich.ac.uk (G.S.)

Received: 31 October 2020; Accepted: 08 December 2020; Published: 14 December 2020

Abstract: Information privacy is a critical design feature for any exchange system, with privacy-preserving applications requiring, most of the time, the identification and labelling of sensitive information. However, privacy and the concept of "sensitive information" are extremely elusive terms, as they are heavily dependent upon the context they are conveyed in. To accommodate such specificity, we first introduce a taxonomy of four context classes to categorise relationships of terms with their textual surroundings by meaning, interaction, precedence, and preference. We then propose a predictive context-aware model based on a Bidirectional Long Short Term Memory network with Conditional Random Fields (BiLSTM + CRF) to identify and label sensitive information in conversational data (multi-class sensitivity labelling). We train our model on a synthetic annotated dataset of real-world conversational data categorised in 13 sensitivity classes that we derive from the P3P standard. We parameterise and run a series of experiments featuring word and character embeddings and introduce a set of auxiliary features to improve model performance. Our results demonstrate that the BiLSTM + CRF model architecture with BERT embeddings and WordShape features is the most effective (F1 score 96.73%). Evaluation of the model is conducted under both temporal and semantic contexts, achieving a 76.33% F1 score on unseen data and outperforms Google's Data Loss Prevention (DLP) system on sensitivity labelling tasks.

Keywords: BiLSTM; BERT; NLP; context-aware

1. Introduction

Diminishing information privacy in communication ecosystems often requires consumers to directly manage their own preferences. Yet, this easily becomes a tedious task considering the amount of information exchanged on a daily basis. Research indicates that, most often, consumers lack the information to take appropriate privacy-aware decisions [1], and even where sufficient information is available, long-term privacy is traded-off for short-term benefits [2] (indicatively, a Facebook-focused empirical user study pointed out that 20% of participants and 50% of interviewees reported direct regret after posting sensitive information publicly) [3]. Automatically identifying sensitive information is critical for privacy-preserving technologies. Sensitive data most often appear as unstructured text, making it vulnerable to privacy threats, due to its parseable and searchable format. This work solely concentrates on text data.

The sensitivity of a piece of information is directly shaped by the context it is provided in. For the purposes of this analysis, we offer a taxonomy of four distinct context classes, for use as sensitivity categories. We derive these respectively by the meaning, interaction, precedence, and preference associated with any piece of information:

Semantic Context: formed on the basis of the semantic meaning of a term. As, for instance, in the case of homonyms or homographs, the semantic meaning of a sequence affects its sensitivity.

Agent Context: depending upon the agents participating in the transmission of information. Here, the relationship between participating actors determines sensitivity. For instance, patient–doctor sharing of medical history is non-sensitive for this set of actors, but sensitive otherwise.

Temporal Context: defined by information precedence that affects the significance of a term. Here, previously introduced definitions of a term qualify its sensitivity. If, for example, a string sequence is introduced as a password, it carries higher sensitivity than if it was introduced as a username.

Self Context: defined by the user's personal privacy preferences. Notably, what is considered private varies across users due to cultural influences, personal experiences, professional statute, etc. For example, ethnic origin may be considered sensitive information for one individual but not for another.

The above list of context classes is not exhaustive, nor are these mutually exclusive. One or more contexts may simultaneously influence sensitivity differently. Information may, however, be sensitive, regardless of context (e.g., credit-card numbers, email, passwords, insurance numbers). In this work, we use sensitive information as any coherent sequence of textual data from which a third party can elicit information that falls under the categories proposed in the P3P standard (Table 1).

Table 1. Sensitivity classes used for multi-class classification. (Source: P3P Standard [4]) and the count of sensitive tokens in our dataset per class.

Sensitivity Class	Description	Label	Count
Physical Contact Information	Information that allows an individual to be contacted or located in the physical world—such as telephone number or address.	PHSCL_CNCT_INFO	14,108
Online Contact Information	Information that allows an individual to be contacted or located on the Internet—such as email. Often, this information is independent of the specific computer used to access the network. (See the category "Computer Information")	ONLINE_CNCT_INFO	30,248
Unique Identifiers	Non-financial identifiers, excluding government-issued identifiers, issued for purposes of consistently identifying or recognising the individual. These include identifiers issued by a Web site or service.	UNIQUE_ID	10,594
Purchase Information	Information actively generated by the purchase of a product or service, including information about the method of payment.	PURCHASE_INFO	13,712
Financial Information	Information about an individual's finances including account status and activity information such as account balance, payment or overdraft history, and information about an individual's purchase or use of financial instruments including credit or debit card information.	FINANCIAL_INFO	19,258
Computer Information	Information about the computer system that the individual is using to access the network—such as the IP number, domain name, browser type or operating system.	COMPUTER_INFO	8143
Demographic and Socioeconomic Data	Data about an individual's characteristics—such as gender, age, income, postal code, or geographic region.	DEMOG_SOCECON_INFO	26,013
State Management Mechanisms	Mechanisms for maintaining a stateful session with a user or automatically recognising users who have visited a particular site or accessed particular content previously—such as HTTP cookies.	STATE_MGT	5666
Political Information	Membership in or affiliation with groups such as religious organisations, trade unions, professional associations, political parties, etc.	POLITICAL_INFO	21,341
Health Information	Information about an individual's physical or mental health, sexual orientation, use or inquiry into health care services or products, and purchase of health care services or products.	HEALTH_INFO	23,516
Preference Data	Data about an individual's likes and dislikes—such as favorite colour or musical tastes.	PREFERENCE_INFO	43,048
Location Data	Information that can be used to identify an individual's current physical location and track them as their location changes—such as GPS position data.	LOC_INFO	15,795
Government-issued Identifiers	Identifiers issued by a government for purposes of consistently identifying the individual.	GOVT_ID	14,108

This paper proposes a context-aware predictive deep learning model that can annotate sensitive tokens in a sequence of text data, more formally defined as a "token sensitivity labelling" task. We develop our models for *semantic* and *temporal* contexts, as this provides an adequate proof-of-concept, and as incorporating all four contexts requires extraneous methodologies, which are planned for future work. We reduce the problem of sensitivity annotation to a multi-class classification problem and follow deep learning techniques that were proven effective in similar labelling tasks, such as Named Entity Recognition (NER) and Part-of-Speech (POS) tagging [5–7]. We develop a context-aware classifier based on the BiLSTM + CRF architecture with word embeddings and WordShape as features. We address dataset limitations by developing data generation algorithms to combine synthetic and real data, and experimentally identify the best performing model architecture and feature combination. The models are first evaluated on a dataset generated with the assistance of synthetic sensitive data, reaching an F1 score of 96.73%. We evaluate our model in two settings: (i) one addressing sensitive information annotation under the influence of temporal context and; (ii) one comparing against Google's DLP system with semantic context variations. In the former our model reaches an F1 score of 76.33%, and in the latter, results highlight the resiliency of our system on semantic noise by outperforming Google's DLP in all sensitive information type annotation. We summarise the key contribution of this work as:

1. The introduction of the four context classes (Semantic, Agent, Temporal, Self) as a taxonomy to suitably represent the relationship between candidate sensitive tokens and their textual surroundings (sentence or wider textual sequence). In addition to allowing for context differentiation (for instance, depending on the *nature* of privacy settings), the taxonomy may be used as a framework for applying sensitivity tiers (for instance by overlaying different types of context-awareness depending on the *strictness* of privacy settings).
2. The use of BiLSTM-CRF multi-class annotation of sensitive tokens in a *context-aware* manner, (an approach that, to the best of our knowledge has scarce representation in the literature), the implementation of which outperforms an industry-strength system in sensitivity labelling along at least one context class. (More specifically, our approach differentiates from normal NER (Named-Entity Recognition) tagging in two ways: (a) Although it is technically still a sequence labelling task, it is fundamentally different in the same way that CWI (Complex Word Identification) tagging differs from NER. Our sensitivity labelling is context-aware whereas NER is not. (b) Our work includes a data generation strategy, something that would not be needed in the case of NER).
3. A dataset enrichment methodology to address the scarcity of public annotated data with sensitivity labelling, which uses real-world conversational data as seeds to generate a large-enough training set while mitigating any sensitivity class imbalances.

Notably, our work aims to investigate the performance of our BiLSTM + CRF architecture for context-aware token sensitivity labelling, as opposed to discovering the optimum model for the task; this is clearly reflected in the experimental design where variants of that model are evaluated in temporal and semantic contexts. In the absence of similar research for more complex architectures, comparisons with other similar BiLSTM-based architectures are performed to initially investigate the behaviours of simpler models.) The paper is organised into eight sections: Section 2 first provides the background and related work, followed by a discussion of our methodological approach and model background (Section 3); we then devote Section 4 to our dataset creation strategy, and follow up with our experimental design (Section 5). Implementation and results are outlined in Section 6, and our model is then evaluated (Section 7). A discussion on our contributions, limitations and future work is ultimately offered in Section 8.

2. Related Work

The majority of the literature on sensitivity labelling is associated with Data Loss Prevention (DLP) systems [8–12], notably focusing on classifying sensitivity at the document level. Other research uses sensitivity classification for confidential information redaction on declassified documents [13–15], where classification is often performed at finer granularity that reaches the token level. More general applications include quantifying information leakage in Open Social Networks (OSNs) for privacy-preserving technologies [16,17].

Earlier work on text sensitivity annotation focused on heuristics. *Sweeney* [18] introduced a template matching approach with boolean hashtables to capture Personally Identifiable Information (PII) in medical records with 99–100% accuracy; however, this approach is challenging when dealing with unstructured data and requires manual work to be expanded to other domains. Gomez-Hidalgo et al. [11] used NER to pinpoint sensitive tokens in a corpus. Although their assumptions about the sensitive nature of Named Entities are reasonable, static NER is context-free and only captures a limited part of sensitive content. Sanchez et al. [19] presented an information-theoretic approach by introducing the concept of information content (IC), defined later in Section 3.4. They annotated as sensitive any noun-phrase with an IC value higher than a threshold β. Again, this is a context-free approach and using only IC as a sensitivity measure is problematic in some cases, as it is directly related to the size and content of the corpus used.

A different approach for text sensitivity annotation uses statistical machine learning models. The work of Hart et al. offered a DLP system, which can classify sensitive enterprise documents using a Support Vector Machine (SVM) classifier trained on a WikiLeaks-based corpus [10]. Later, MacDonald et al. built a novel SVM sensitivity classifier by mixing concepts from both NLP and machine learning for government document declassification, using POS n-gram tags as a sensitivity load indicator [15]. Alzhrani et al. [9] proposed another DLP system with more fine-grained granularity. Their work effectively combined unsupervised and supervised methods to create a similarity-based classifier operating on a paragraph level and trained on an ad-hoc annotated WikiLeaks corpus. Building on their previous work, MacDonald et al. further enhanced their SVM classifier by introducing pre-trained Word2Vec and GLoVe word embeddings [20,21] and found that word embeddings can significantly contribute to a more accurate model. Our work is the closest to their approach.

Research using deep learning for textual sensitivity annotation is relatively sparse, with only a few authors moving to that direction. Ong et al. built a context-aware DLP system, which follows a hierarchical structure to achieve fine-grained granularity [8], and used LSTM neural networks to achieve binary sensitivity classification at the token level. Despite the novelty of their hierarchical approach, we argue that the dataset size used for the experiments may fall short of the requirements for deep learning applications. Jiang et al. [17] used LSTM networks for identifying personal health experiences from tweets. Similarly, previous work underlines the high utility of word embeddings and LSTM/BiLSTM networks for textual data mining and classification through social media posts and other sources [22–24]. Although these are framed in other problem domains, their results highlight the advantages of deep learning methodologies against conventional machine learning models for similar tasks.

3. Background and Approach

We assess the performance of specific BiLSTM variants in classifying and labelling information sensitivity in a particular context. Token relationships in this problem definition are sequential; therefore, we focus on sequential models such as BiLSTM and CRF, with supervised training. We chose a bidirectional LSTM, over simple LSTM, for better modelling of temporal nuances in a corpus, as BiLSTMs perform forward and backwards passes on sequential data and model data dependencies in both directions. Finally, we introduce auxiliary features, namely, POS tags, Information Content (IC) and WordShape (WS).

3.1. LSTM Networks

First, we define the BiLSTM Recurrent Neural Network (RNN) more formally. A recurrent neural network is a special type of normal artificial neural network (ANN) which is capable of modelling sequential data by having recurrent connections [25]. In essence, it maintains a hidden state, which can be considered as a "memory" of previous inputs. This is driven by the fact that each neuron represents an approximation function of all previous data.

Figure 1 illustrates the architecture of a simple RNN. The input units $\{\ldots, x_{t-1}, x_t, x_{t+1}, \ldots\}$ where $x = (x_1, x_2, x_3, \ldots, x_N)$, are connected to the hidden units $h_t = (h_1, h_2, \ldots, h_M)$ in the hidden layer, via connections defined by weight matrix W_{IH}. Every hidden unit is connected to the next one with recurrent connections given by W_{HH}. Each hidden unit is therefore formulated by:

$$h_t = f_H(o_t) \tag{1}$$

where:

$$o_t = W_{IH} + W_{HH}h_{t-1} + b_h \tag{2}$$

F_h is a non-linear function such as tanh, ReLU or sigmoid, etc., and b_H is the bias vector. The hidden layer is also connected with the output layer with weights W_{HO}. Lastly the outputs $y_t = (y_1, y_2, \ldots, y_P)$ are defined by:

$$y_t = f_O(W_{HO}h_t + b_o) \tag{3}$$

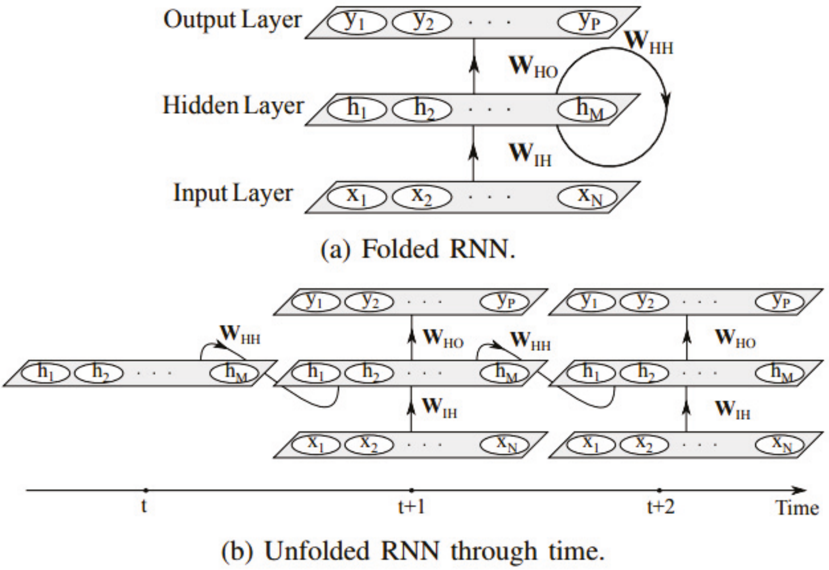


Figure 1. Simple Recurrent Neural Network (RNN) architecture. For the sake of simplicity biases are ignored. Source: [26].

In the same manner as the hidden layer, f_O is the activation function and b is the bias vector.

Although, this model maintains a memory of previous states, in practice it suffers from the vanishing gradient problem, thus becoming impractical for long-term dependencies [27]. A special type of RNN called Long Short Term Memory (LSTM) was published in 1997, which overcomes this issue [28]. LSTM cells follow a more sophisticated mechanism with the introduction of a complex cell that utilises "forget" gates to selectively choose what to forget. An illustration of the LSTM is given in Figure 2.

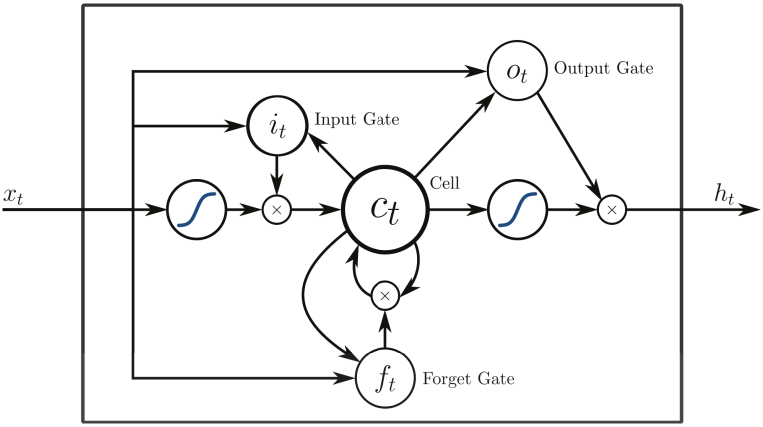


Figure 2. Long Short Term Memory (LSTM) unit. For the sake of simplicity biases are ignored.

The state of an LSTM memory unit adopts the following mathematical formulation:

$$\begin{aligned}
i_t &= \sigma(W_{xi} + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \otimes tanh(c_t)
\end{aligned}$$

To clarify, the subscripts correspond to the initials of what each matrix represents (i.e., W_{hf} is the hidden forget weight matrix). Additionally, f, i, o and c correspond to the forget, input, output and cell gate vectors. With these in mind, an LSTM network would resemble the initial RNN architecture provided above but with LSTM cells instead.

However, due to its architecture an LSTM network can only perform forward passes on sequential data, which ultimately means that the data dependencies are only modelled uni-directionally. An intuitive way to overcome this limitation is to use an exact replica of the LSTM network but in reverse. Thus, combining these two together, a Bidirectional LSTM (BiLSTM) is created which can be used to model dependencies bidirectionally.

3.2. Embeddings

Embeddings provide an efficient mechanism for encoding the semantic and temporal context information. They were proven very effective in practical neural network applications for encoding complex data structures to information-rich continuous vectors in latent space. Examples include mappings from word to vectors [20,21,29], document to vectors [30], graphs to vectors [31], etc. Especially word vectors have become the norm in neural networks for Natural Language Processing (NLP). We address two main types of embeddings: context-free and contextual. Context-free embeddings are static and do not capture any context about the word. Conversely, a contextual embedding encapsulates information about the surrounding context of the word, hence assigning embedding vectors v_i for each of the i contexts in the embedding space.

3.3. Conditional Random Fields

Conditional Random Fields (CRFs) is a discriminative model mostly used for labelling and separating sequential data [7]. The underlying concept of CRFs attempts to model a conditional probability distribution over a label sequence given an already observed sequence. The use of

Conditional Random Fields (CRFs) in this work is inspired by state-of-the-art results of Neural CRFs [32] particularly combined with BiLSTMS in fundamental sequence labelling NLP tasks such as NER and POS tagging [6,33]. Neural CRFs implement neural networks to extract high-level features for use as inputs to CRFs for labelling. CRFs' architecture models the conditional distribution P(x | y) over a label sequence, given an already observed sequence, rather than the joint distribution P(x,y), thus outperforming traditional machine learning models, such as Hidden/Maximum Entropy Markov Models (HMMs, MEMMs) [7], in sequential labelling.

3.4. Auxiliary Features (POS, IC, WS)

Although relying solely on word embeddings for prediction can deliver acceptable performance, we propose three additional features to enrich the learning significance of the semantic and syntactic abstraction of word embeddings: POS tags, Information Content (IC) and WordShape (WS).

MacDonald et al. [15] showed that specific POS n-gram sequences can be correlated with sensitive information. This provided an incentive to use POS tags as auxiliary features in this work. We extract the POS tags of the sequences using SpaCy's v2.1 (https://github.com/explosion/spaCy) POS tagger, preferred for its speed, industrial strength and convenience.

Information Content (IC) offers a quantitative metric for general purpose sensitivity [19]. IC estimates the information carried by a specific token *t* in a given context, relying on the information-theoretic assumption that rare terms typically convey more information than general terms (e.g., "surgeon" vs. "doctor"). Thus, we expect that incorporating IC in our model can better classify sensitive tokens of particular classes. An example is labelling passwords: due to their random nature, the embeding of a password will most often result in the Out-Of-Vocabulary (OOV) embedding. Hence, apart from the surrounding context, there is nothing differentiating it from other OOV tokens. With this in mind, introducing IC features can be advantageous. As this, however, directly depends on the size and content of the corpus used to calculate the information content, a massive general corpus is required for general purpose estimations. To extract the IC of tokens we used the Bing search engine API (https://azure.microsoft.com/en-gb/services/cognitive-services/bing-web-search-api/). Notably, Google could be more accurate, since it maintains the largest and most updated page index to date. Yet, it was not possible to use Google's search API for this project due to its API restrictions on repetitive use. As suggested by Sanchez et al. [19] only nouns are queued for the IC extraction as the rest of the part-of-speech types have a dynamic meaning that effectively makes search engine queries unreliable for IC calculation. These tokens are assigned an IC value of 0 by default.

Lastly, a morphological word feature, *WordShape*, was introduced in our experiments. We use the term *WordShape* as the textual representation of a word's morphology, which is implemented through transforming words into character sequence templates. This can contribute to learning the sensitivity correlations with structured data such as credit-card numbers, national insurance numbers, and phone numbers. To generate the WordShape features, SpaCy's v2.1 parser was used (https://spacy.io/usage/linguistic-features).

4. Dataset Creation

Public annotated datasets for token sensitivity labelling are rare. Even where such data can be collected, the subjects' privacy is at risk through deanonymisation [34]. We therefore generate synthetic data for training purposes. Training deep learning models on synthetic data often comes with generalisability challenges due to overfitting, although recently, several scholars successfully trained such models on synthetic data in real-world settings [35]. In this paper, we developed a methodology (Section 4.2) to generate a large-enough synthetic annotated dataset, combining real-world conversational data with random sensitive information. At the highest level, two data generation approaches are used: (1) One featuring sensitivity classes that are redacted by default and (2) one with sensitivity classes that are not redacted by default. For the former, consider the data generation process for the "Online contact information" sensitivity class. For example, we choose "email" for the topic. We populate a list of conversational patterns relevant to that topic, ('my email . . . ', 'You can contact me at', . . .) and use that list to search in Reddit threads using Google's BigQuery. Then, we generate a synthetic concrete value for the sensitive part (i.e., the email address), combine it with the conversational pattern used for the query and inject it into the results as part of the conversation (comments chain). Because Google BigQuery redacts sensitive terms, we cannot ascertain their original position. Hence, the modified sentence arrays are injected at a random index. In the latter case (sensitivity classes not redacted by default) we follow a slightly different approach. Again, we use the related conversational pattern to search in Reddit threads, but this time the sensitive part is already included in the comments after the pattern. Therefore, we generate concrete values beforehand and include them in the filtering process. For instance, the "Religion" topic has a pattern "I believe in" and concrete values "Christianity", "Buddhism", etc., which are used as part of the query. Then we annotate the position of the sensitive concrete value and expand it until the next verb or noun is found in the sentence. Lastly, we annotate the conversational pattern "I believe in" along with the next noun. This is not a fail-proof methodology but it generated an acceptable format of the dataset that is tested in our experiments. Our synthetic datasets are derived from real conversational data from Reddit. Where used, manual annotations are part of the training process, in the same way as, for instance, human annotation in object recognition. With this approach, we increase the size of the dataset with much lower effort than that for gathering more real-world data, and we also mitigate sensitivity class imbalance [36–38]. We then test our proposed dataset on real unseen data.

4.1. Sensitivity Classes

To increase the semantic significance of our sensitivity classification we used 13 distinct sensitivity classes, based on the categories specified in W3C's Platform for Privacy Preferences (P3P) [4] presented in Table 1. Note that the P3P specification originally defined 17 categories but we intentionally omitted 4 as they were very open-scoped; These are *Navigation and Click-stream Data*, *Interactive Data*, *Content* and *Other*. This made it easier to automate the aggregation of data for unambiguously defined sensitivity classes for dataset creation. It also allowed to examine model performance for each sensitivity class individually, and potentially extract more relevant insights. The choice to use the W3C P3P classes was made to (i) leverage existing legal and social expertise that informed the development of the platform, and (ii) support the openness and extensibility of our methods by allowing third-party user applications to be built on top of our work. Since P3P works with other standardised languages, such as APPEL (a P3P Preference Exchange Language), a third party automated process can use this language to trigger an action on leakage of annotated sensitive data.

4.2. Data Generation Process

For our synthetic data generation, we extracted real-world text data from a main discussion theme and then injected sensitive data at random positions. For dataset creation purposes sensitivity classes are further divided into relevant *topics*, thus achieving higher granularity: for instance, the *Financial Information* sensitivity class would include topics like *Payment History*, *Credit Card Numbers*, *Account Balance*, etc. The process was repeated per topic in each sensitivity class. Because the source used for the real-world data redacts sensitive information by design, the injected text was automatically annotated as sensitive and the rest of the corpus as non-sensitive. To cover still for sensitivity classes that leak secondary private information (e.g., *Preference*), we developed an alternative algorithm for annotating such, not so obvious, sensitive data. A very high level flow chart of both algorithms is shown in Figure 3.

Note that the algorithms require three distinct datasets: (i) the sensitive text patterns of the topic (also used to query the data in the first place); (ii) the concrete values and; (iii) the conversational data associated with the topic. To aggregate real conversational data from the web, we utilised the publicly available Reddit comments dataset as given by Google's BigQuery (https://bigquery.cloud.google.com/

dataset/fh-bigquery:reddit_comments). The rich querying capabilities of BigQuery allowed convenient filtering for specific topics by relevant keywords.

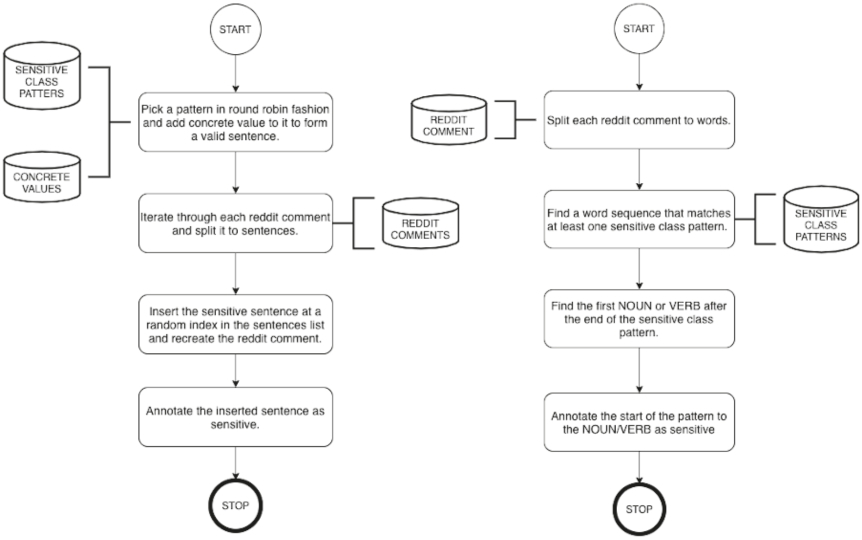

START
SENSITIVE CLASS PATTERS
CONCRETE VALUES
Pick a pattern in round robin fashion and add concrete value to it to form a valid sentence.
Iterate through each reddit comment and split it to sentences.
REDDIT COMMENTS
Insert the sensitive sentence at a random index in the sentences list and recreate the reddit comment.
Annotate the inserted sentence as sensitive.
STOP
START
REDDIT COMMENT
Split each reddit comment to words.
Find a word sequence that matches at least one sensitive class pattern.
SENSITIVE CLASS PATTERNS
Find the first NOUN or VERB after the end of the sensitive class pattern.
Annotate the start of the pattern to the NOUN/VERB as sensitive
STOP

Figure 3. Synthetic data generation (i) when sensitive data is redacted by default (**left**) and (ii) when it is not (**right**).

Figure 3 (left), outlines the algorithm for augmenting our dataset with artificially created sentences that include at least one sensitive term. A sensitive term is encoded as a unique token, and then replaced with a concrete value. Concrete values are the actual mock values that make the sentence sensitive, for example, passwords, email, address These were randomly generated using Mockaroo (https://mockaroo.com/), a generation engine for realistic data.Then, the Reddit comments are split into sentences and the newly created sensitive sentence is injected at a random index within the sentence array. That sentence is annotated as sensitive and the remaining sequence as non-sensitive. The algorithm in Figure 3 (right) annotates sensitive information already present in Reddit comments. We selectively built a collection of phrases that correspond to a topic in a sensitivity class and then used these to query the Reddit comments. The phrases are then matched per comment, and all tokens from the matching phrase until the next verb or noun, are annotated as sensitive.

After data generation, we built an annotated dataset of multiple sensitivity classes. Table 1 shows the number of annotated tokens per sensitivity class in the dataset. The classes are notably imbalanced because an upper bound constraint on how many records can be generated per topic was introduced. While this constraint was imposed to avoid overfitting the model to one specific topic, the observed imbalance is realistic as it is often seen in real-world datasets [39]. Overall, 12% of the total tokens in the dataset are annotated as sensitive, with the remaining 88% labelled as non-sensitive.

5. Experimental Design

Our experiments attempt to answer three questions: (i) Which of our BiLSTM + CRF model architectures and word embeddings combination is better suited for the task; (ii) whether character embeddings contribute to increased accuracy on the model; and (iii) whether IC, POS tags and WordShape features increase model performance. For that purpose, 11 distinct BiLSTM + CRF model variations were derived for experimentation, as shown in Table 2. In addition, five simpler models were selected for bench marking against the main BiLSTM-CRF variants.

We chose the hyperparameters of the models based on empirical results [40] and preliminary experiments. The input sequences were trimmed to 205 timesteps (maximum sequence length in the dataset) and 128 units of BiLSTM cells were used, followed by a dropout layer of 40%. For experiments with character embeddings, a character input sequence of 32 length was used in a 1D convolutional layer with a kernel of size 3, followed by a dropout layer of 50% and a Global Max Pooling layer. Training was performed in batches of 128 for 100 epochs but an early stopping callback was employed to interrupt training if validation loss was not improved for 5 consecutive epochs.

The test subset consists of 10% of the dataset and was entirely left out for use as a completely unbiased evaluation dataset. The remaining 90% was further split to 80% training (used to fit the classifier weights), and 20% validation data for tuning the hyperparameters. For preprocessing, the text was converted to lowercase, all contractions were reversed and punctuation removed. Unlike conventional preprocessing pipelines, where a stopword removal stage is involved, we decided to keep the stopwords, as they are part of our automated annotation process when creating the dataset.

EXPERIMENT A: ***Model design and word embeddings:*** Interestingly, there is a wide range of BiLSTM applications in NLP often featuring state-of-the-art results [41–43]. Similarly, the integration of BiLSTM with a CRF layer is also rapidly gaining research attention and has delivered promising results in NLP tasks [6,33,44]. For the above reason, Experiment A focuses on reviewing the performance effect of word embeddings on BiLSTM + CRF models (see EXP. A5–A7 in Table 2). Four alternative variants (EXP. A1–A4) were also used, to offer a basis for comparison.

Even though existing literature has demonstrated the advantages of using contextualised word embeddings in numerous occasions [45–47], we perform this experiment to support this hypothesis for this problem setting as well. Of the many available word embedding extraction techniques [48] we shortlisted 3 methods for the evaluation, as a sufficient minimum to cover for all pre-training and contextualisation possibilities. The first uses initially random vectors to derive embeddings in the training process, and is here referred to as Randomised Word Embedding (RWE). The vocabulary of the RWE embeddings was built on the training split of our dataset. RWE is later used as a baseline for comparing with pre-trained word embeddings. The remaining two choices were pre-trained word embeddings, namely GLoVe and BERT [5,21] to cover context-free and contextualised embeddings, respectively. GLoVe is a popular context-free word embeddings model and BERT comes from Google's BERT, a language model that achieved state-of-the-art performance in many standard NLP tasks. In the case of BERT, the output of the last encoder layer was used as embeddings.

EXPERIMENT B: ***Character embeddings:*** Character-level embeddings were successfully combined with word embeddings to improve performance before [49,50]. As the integration of character embeddings allows for learning language-agnostic morphological features, we attempt to quantify the resulting performance improvement, if any.

It has been shown that LSTM and Convolutional Neural Network (CNN) character embeddings exhibit similar performance improvements when combined with BiLSTM models, with a slight advantage for CNN [51]. We implement character embeddings through an additional extension model based on a one-dimensional convolution layer. The output of the extension model is concatenated with the input of the best-performing embedding types.

EXPERIMENT C: ***Auxiliary features experiment:*** Section 3.4 provides a detailed account of the three auxiliary features (POS, IC, WS) used to enhance model performance, and also articulated sources and selection strategies. In the experimental setting, we introduce 7 feature variations on top of the best performing BiLSTM + CRF model architecture. The aim is to practically evaluate the contribution of these features (and their combination) on performance. The variations are illustrated in Table 2 (bottom).

Table 2. Model variations and performance metrics of the experiments. (Top Section): Conditional Random Fields (CRF) layer and embeddings combination results. (Middle Section): Character embeddings model extension experiment results. (Bottom Section): Auxiliary features variations experiment results.

		MODEL			EMBEDDINGS			AUX. FEATURES			RESULTS		
		BiLSTM	CRF	CNN	RWE	GLoVe	BERT	POS	IC	WS	Precision	Recall	F1
EXP. A	1		X								0.8068	0.5779	0.6581
	2	X			X						0.8986	0.9315	0.9147
	3	X				X					0.9347	0.9599	0.9471
	4	X					X				0.9452	0.9689	0.9569
	5	X	X		X						0.9113	0.9337	0.9224
	6	X	X			X					0.9451	0.9558	0.9504
	7	X	X				X				0.9568	0.9693	0.9630
EXP B.	8	X		X			X				0.9568	0.9576	0.9572
	9	X	X	X			X				0.9634	0.9558	0.9596
EXP C.	10	X	X				X	X			0.9548	0.9685	0.9616
	11	X	X				X		X		0.9611	0.9618	0.9614
	12	X	X				X			X	**0.9640**	**0.9706**	**0.9673**
	13	X	X				X	X	X		0.9639	0.9625	0.9631
	14	X	X				X		X	X	0.9635	0.9629	0.9632
	15	X	X				X	X		X	0.9628	0.9615	0.9622
	16	X	X				X	X	X	X	0.9611	0.9704	0.9657

6. Results

Micro-averaged Precision, Recall and F1 metrics have been used for performance evaluation as they are widely used in similar sequence labelling tasks [52–54] and perform better on imbalanced datasets [55]. Table 2 summarises the results for the entire set of experiments carried out, separated in three sections, with the corresponding models and their performance.

EXPERIMENT A: ***Model design and word embeddings:*** As a baseline, a CRF model with casing and word morphology features was used. Table 2 shows that all of our proposed models outperform the baseline CRF. Of these, predictably, RWE performs the poorest. GLoVe embeddings deliver a substantial improvement, with BERT embeddings giving the best results across all three metrics, most likely due to its contextualised nature. On aggregate, results indicate that adding a CRF layer improves the performance of all models slightly. In summary, although the increase of the F1 metric is very marginal, there is a consistent improvement throughout all variants in experiment A when introducing a CRF layer. We observe that $\text{BiLSTM}_{\text{BERT}} + \text{CRF}$ is the best performing CRF-enriched model in regards to model architecture and embeddings.

EXPERIMENT B: ***Character embeddings:*** Despite our expectation for the contrary, results revealed that character embeddings cause performance deterioration. Reasons for this may be: (a) that the supplementary trainable parameters increased the model's complexity and learning the underlying correlations between the data points became more challenging, and (b) that the CNN and pooling architecture is perhaps by design unsuitable for this problem setting. The remaining experiments were conducted without CNN character embeddings.

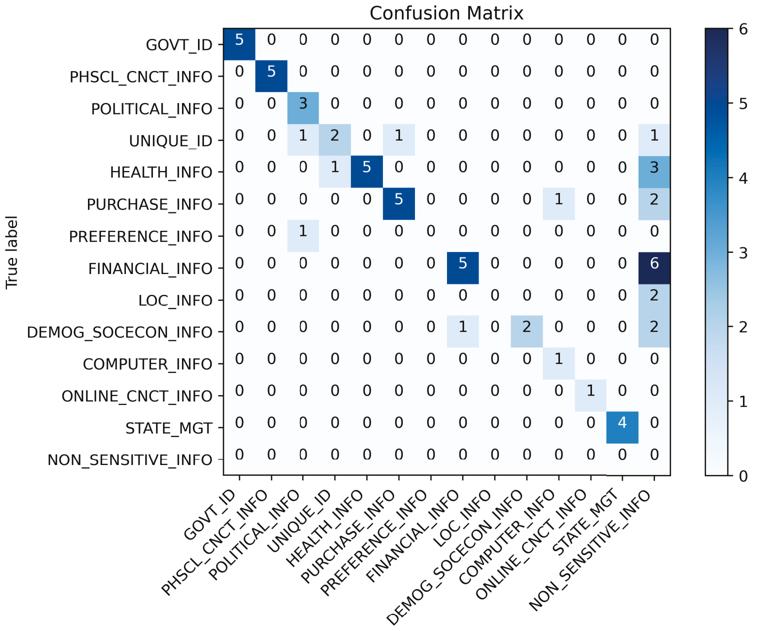
EXPERIMENT C: ***Auxiliary features experiment:*** For the third part of the experiment we used POS tags, IC and WordShape as auxiliary features for the classification. Overall, it is observed that the combination of two or more auxiliary features offers a slight performance advantage against single feature models. Yet, the performance metrics when using those features are not dissimilar from the initial BiLSTM + CRF model with BERT embeddings, except when using WordShape features exclusively. Thus, based on the Recall and F1 metrics, we identify WordShape as a better suited auxiliary feature that can be used with BiLSTM + CRF model. Accordingly, we chose to incorporate WordShape features for further evaluation experiments.

7. Evaluation

Based on the experimental results presented in Section 6 we evaluate BiLSTM$_{\text{BERT+WS}}$ + CRF against temporal and semantic context sensitivity labelling.

For temporal context sensitivity labelling, the final model is evaluated on a dataset that was manually built and annotated. Manual annotation is time-consuming and thus the dataset is small compared to the synthetic one. It consists of 60 text sequences, specifically written in a way that token sensitivity is directly dependent on temporal context.

Due to manual annotation, there were cases where stopwords were annotated as sensitive tokens but not picked up by the model, or vice versa, causing a drop in the evaluation metrics. A confusion matrix (Figure 4) on class-level (rather than token level) granularity offers a better evaluation that is invariant to annotation discrepancies. In effect this demonstrates whether the model managed to identify the sensitivity class of the text. Figure 4 illustrates that the majority of sensitivity types are classified correctly. Most of the incorrect classifications are confused with the *NON_SENSITIVE* class. Note that the hardest classes to classify are the *DEMOG_SOCECON_INFO* and *POLITICAL_INFO*. Additionally, it is important to highlight that the 76.33% and 73.07% F1 score in token-level and class-level experiments, respectively (Figure 4), support our hypothesis that a BiLSTM model can be used for temporal context sensitivity annotation.



	GOVT_ID	PHSCL_CNCT_INFO	POLITICAL_INFO	UNIQUE_ID	HEALTH_INFO	PURCHASE_INFO	PREFERENCE_INFO	FINANCIAL_INFO	LOC_INFO	DEMOG_SOCECON_INFO	COMPUTER_INFO	ONLINE_CNCT_INFO	STATE_MGT	NON_SENSITIVE_INFO
GOVT_ID	5	0	0	0	0	0	0	0	0	0	0	0	0	0
PHSCL_CNCT_INFO	0	5	0	0	0	0	0	0	0	0	0	0	0	0
POLITICAL_INFO	0	0	3	0	0	0	0	0	0	0	0	0	0	0
UNIQUE_ID	0	0	1	2	0	1	0	0	0	0	0	0	0	1
HEALTH_INFO	0	0	0	1	5	0	0	0	0	0	0	0	0	3
PURCHASE_INFO	0	0	0	0	0	5	0	0	0	0	1	0	0	2
PREFERENCE_INFO	0	0	1	0	0	0	0	0	0	0	0	0	0	0
FINANCIAL_INFO	0	0	0	0	0	0	0	5	0	0	0	0	0	6
LOC_INFO	0	0	0	0	0	0	0	0	0	0	0	0	0	2
DEMOG_SOCECON_INFO	0	0	0	0	0	0	0	1	0	2	0	0	0	2
COMPUTER_INFO	0	0	0	0	0	0	0	0	0	0	1	0	0	0
ONLINE_CNCT_INFO	0	0	0	0	0	0	0	0	0	0	0	1	0	0
STATE_MGT	0	0	0	0	0	0	0	0	0	0	0	0	4	0
NON_SENSITIVE_INFO	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1
Token-Level	0.8057	0.7252	0.7633
Class-Level	0.6333	0.8636	0.7307

Figure 4. Confusion matrix for temporal context evaluation.

For our Semantic Context Evaluation, we performed a comparative evaluation against Google's DLP system (https://cloud.google.com/dlp/), which provides industrial-strength sensitive data annotation for over 80 sensitive data types. Google DLP is chosen for its industrial strength and was seen as a suitable benchmark for semantic evaluation since it also uses an automated methodology.

To perform an as accurate as possible comparison, we test the performance of the two systems (our model and DLP) solely on sensitive data types, which are common between the two. Google DLP is provided as a platform with standard functionality, allowing solely for user intervention in (i) selecting InfoTypes (equivalent to *topics* of our Sensitivity Classes) and; (ii) creating templates to support a more structured data detection approach. To that end, another dataset was created manually, containing text sequences that include sensitive tokens affected by semantic context (as an example of a sample in this dataset, consider the sentences "I am a male" and "I am a man", they both reveal the same gender information but the wording is different). In the absence of sizeable overlap between the two systems on sensitivity topics, we select only those commonly appearing in both. These are: *Email, Credit Cards, Age, IP, SSN, Ethnic Group and Gender*.

Results of our comparison are provided in Figure 5. For all sensitive data types, our system outperforms Google's DLP service. Particularly it was observed that, when noise that can affect the syntactic but not the semantic meaning of sensitive data was added, Google's DLP fails to annotate the sensitive tokens. On the contrary, our system exhibits resilience against such noise with a relative accuracy advantage over the Google DLP of 42.65%.

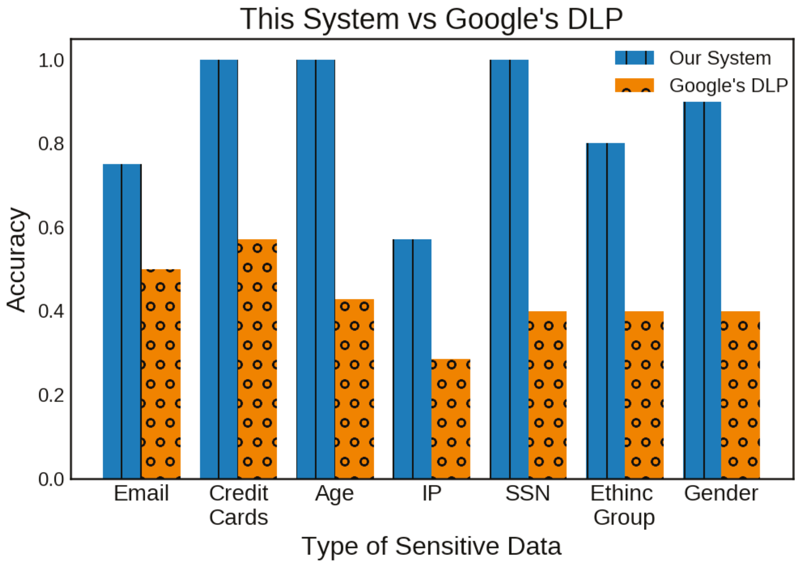


Figure 5. Comparative evaluation between our system and Google's Data Loss Prevention (DLP) system.

8. Discussion

Sensitive information labelling is a prominent problem when designing privacy-aware decision systems. Automated sensitivity labelling is especially relevant when considering users as custodians of their own personal data. With this in mind, we developed our model to enhance sensitivity annotation by first offering a taxonomy of four context classes (semantic, agent, temporal and self), and then using these to implement context-aware labelling.

Our model does not come without limitations, and future work should: involve the full set of context classes; extend the auxiliary features experiments to evaluate tweaked models such as plain BiLSTM (without CRF); incorporate additional word, sentence and character embeddings; and perform testing and validation on more extensive datasets. Yet, the work presented in this paper can essentially serve as a framework for building similar models within alternative well-defined problem domains.

The impact of our work is manifold, although we acknowledge potential risks that typically come with advancements in understanding and extracting sensitive information. While our models contribute to more accurate context-aware sensitivity labelling, our choice to adopt P3P sensitivity classes also supports openness and extensibility to third party applications, offering a platform for others to further develop suitable methods. It furthermore demonstrates promising results from using deep learning techniques in text sensitivity annotation, an area that is sparsely addressed in the literature. We believe that further expanding our approach will offer a more concrete future direction for privacy-preserving information exchange.

Author Contributions: This paper was accomplished based on the collaborative work of the authors. A.P. performed the experiments and analysed the data. Experiment interpretation and paper authorship were jointly performed by A.P. and G.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Lond-Short Term Memory
CNN	Convolutional Neural Network
CRF	Conditional Random Field
CWI	Complex Word Identification
DLP	Data Loss Prevention
IC	Information Context
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
NER	Named Entity Recognition
OOV	Out-of-Vocabulary
POS	Part of Speech
SVM	Support Vector Machine
WS	WordShape

References

1. Acquisti, A.; Adjerid, I.; Balebako, R.; Brandimarte, L.; Cranor, L.F.; Komanduri, S.; Leon, P.G.; Sadeh, N.; Schaub, F.; Sleeper, M.; et al. Nudges for privacy and security: Understanding and assisting users' choices online. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–41. [CrossRef]
2. Acquisti, A.; Grossklags, J. Privacy and rationality in individual decision making. *IEEE Secur. Priv.* **2005**, *3*, 26–33. [CrossRef]
3. Wang, Y.; Norcie, G.; Komanduri, S.; Acquisti, A.; Leon, P.G.; Cranor, L.F. I regretted the minute I pressed share: A qualitative study of regrets on Facebook. In Proceedings of the Seventh Symposium on Usable Privacy and Security, Pittsburgh, PA, USA, 20–22 July 2011; ACM: New York, NY, USA, 2011; p. 10.
4. Cranor, L.; Dobbs, B.; Egelman, S.; Hogben, G.; Humphrey, J.; Langheinrich, M.; Marchiori, M.; Presler-Marshall, M.; Reagle, J.M.; Schunter, M.; et al. *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*; Note NOTE-P3P11-20061113; World Wide Web Consortium: Cambridge, MA, USA, 2006.
5. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
6. Ma, X.; Hovy, E. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv* **2016**, arXiv:1603.01354.
7. Lafferty, J.; McCallum, A.; Pereira, F.C. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Available online: https://repository.upenn.edu/cis_papers/159/ (accessed on 13 June 2020)

8. Ong, Y.J.; Qiao, M.; Routray, R.; Raphael, R. Context-Aware Data Loss Prevention for Cloud Storage Services. In Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, USA, 25–30 June 2017; pp. 399–406. [CrossRef]
9. Alzhrani, K.; Rudd, E.M.; Boult, T.E.; Chow, C.E. Automated Big Text Security Classification. In Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI), Tucson, AZ, USA, 28–30 September 2016.
10. Hart, M.; Manadhata, P.; Johnson, R. Text classification for data loss prevention. In Proceedings of the 11th International Conference on Privacy Enhancing Technologies, Waterloo, ON, Canada, 24 July 2011; Springer: Berlin/Heidelberg, Germany, 2011.
11. Gomez-Hidalgo, J.M.; Martin-Abreu, J.M.; Nieves, J.; Santos, I.; Brezo, F.; Bringas, P.G. Data leak prevention through named entity recognition. In Proceedings of the 2010 IEEE Second International Conference on Social Computing, Minneapolis, MN, USA, 20–22 August 2010; pp. 1129–1134.
12. Alneyadi, S.; Sithirasenan, E.; Muthukkumarasamy, V. Word N-gram based classification for data leakage prevention. In Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia, 16–18 July 2013; pp. 578–585.
13. McDonald, G.; Macdonald, C.; Ounis, I.; Gollins, T. Towards a classifier for digital sensitivity review. In *European Conference on Information Retrieval*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 500–506.
14. McDonald, G.; Macdonald, C.; Ounis, I. Enhancing sensitivity classification with semantic features using word embeddings. In *European Conference on Information Retrieval*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 450–463.
15. McDonald, G.; Macdonald, C.; Ounis, I. Using part-of-speech n-grams for sensitive-text classification. In Proceedings of the 2015 International Conference on The Theory of Information Retrieval, Northampton, MA, USA, 27–30 September 2015; ACM: New York, NY, USA, 2015; pp. 381–384.
16. Caliskan Islam, A.; Walsh, J.; Greenstadt, R. Privacy detective: Detecting private information and collective privacy behavior in a large social network. In Proceedings of the 13th Workshop on Privacy in the Electronic Society, Scottsdale, AZ, USA, 3 November 2014; ACM: New York, NY, USA, 2014; pp. 35–46.
17. Jiang, K.; Feng, S.; Song, Q.; Calix, R.A.; Gupta, M.; Bernard, G.R. Identifying tweets of personal health experience through word embedding and LSTM neural network. *BMC Bioinform.* **2018**, *19*, 210. [CrossRef] [PubMed]
18. Sweeney, L. Replacing personally-identifying information in medical records, the Scrub system. In Proceedings of the AMIA Annual Fall Symposium 1996, Washington, DC, USA, 30 October 1996; American Medical Informatics Association: Bethesda, MD, USA, 1996; p. 333.
19. Sánchez, D.; Batet, M.; Viejo, A. Detecting sensitive information from textual documents: An information-theoretic approach. In *International Conference on Modeling Decisions for Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 173–184.
20. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*; ACM: New York, NY, USA, 2013; pp. 3111–3119.
21. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
22. Ali, F.; El-Sappagh, S.; Kwak, D. Fuzzy Ontology and LSTM-Based Text Mining: A Transportation Network Monitoring System for Assisting Travel. *Sensors* **2019**, *19*, 234. [CrossRef] [PubMed]
23. Ali, F.; El-Sappagh, S.; Islam, S.R.; Ali, A.; Attique, M.; Imran, M.; Kwak, K.S. An intelligent healthcare monitoring framework using wearable sensors and social networking data. *Future Gener. Comput. Syst.* **2021**, *114*, 23–43. [CrossRef]
24. Ayvaz, E.; Kaplan, K.; Kuncan, M. An Integrated LSTM Neural Networks Approach to Sustainable Balanced Scorecard-Based Early Warning System. *IEEE Access* **2020**, *8*, 37958–37966. [CrossRef]
25. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef] [PubMed]
26. Salehinejad, H.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent advances in recurrent neural networks. *arXiv* **2017**, arXiv:1801.01078.

27. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **1998**, *6*, 107–116. [CrossRef]
28. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
29. Shi, B.; Fu, Z.; Bing, L.; Lam, W. Learning Domain-Sensitive and Sentiment-Aware Word Embeddings. *arXiv* **2018**, arXiv:1805.03801.
30. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1188–1196.
31. Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; Jaiswal, S. graph2vec: Learning Distributed Representations of Graphs. *arXiv* **2017**, arXiv:1707.05005.
32. Artieres, T. Neural conditional random fields. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 177–184.
33. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural architectures for named entity recognition. *arXiv* **2016**, arXiv:1603.01360.
34. Narayanan, A.; Shmatikov, V. How to break anonymity of the netflix prize dataset. *arXiv* **2006**, arXiv:cs/0610105.
35. Emam, K.; Mosquera, L.; Hoptroff, R. *Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data*; O'Reilly Media, Incorporated: Sebastopol, CA, USA, 2020.
36. Hu, G.; Peng, X.; Yang, Y.; Hospedales, T.M.; Verbeek, J. Frankenstein: Learning deep face representations using small data. *IEEE Trans. Image Process.* **2018**, *27*, 293–303. [CrossRef]
37. Das, A.; Gkioxari, G.; Lee, S.; Parikh, D.; Batra, D. Neural Modular Control for Embodied Question Answering. *arXiv* **2018**, arXiv:1810.11181.
38. Patki, N.; Wedge, R.; Veeramachaneni, K. The synthetic data vault. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, 17–19 October 2016; pp. 399–410.
39. Kaur, H.; Pannu, H.S.; Malhi, A.K. A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions. *ACM Comput. Surv.* **2019**, *52*. [CrossRef]
40. Cheng, G.; Peddinti, V.; Povey, D.; Manohar, V.; Khudanpur, S.; Yan, Y. An Exploration of Dropout with LSTMs. In Proceedings of the Interspeech, Stockholm, Sweden 20–24 August 2017.
41. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv* **2018**, arXiv:1804.07461.
42. Talman, A.; Yli-Jyrä, A.; Tiedemann, J. Natural Language Inference with Hierarchical BiLSTM Max Pooling Architecture. *arXiv* **2018**, arXiv:1808.08762.
43. Bohnet, B.; McDonald, R.T.; Simões, G.; Andor, D.; Pitler, E.; Maynez, J. Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings. *arXiv* **2018**, arXiv:1805.08237.
44. Reimers, N.; Gurevych, I. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. *arXiv* **2017**, arXiv:1707.09861.
45. Si, Y.; Wang, J.; Xu, H.; Roberts, K. Enhancing Clinical Concept Extraction with Contextual Embedding. *arXiv* **2019**, arXiv:1902.08691.
46. MacAvaney, S.; Yates, A.; Cohan, A.; Goharian, N. CEDR: Contextualized Embeddings for Document Ranking. *arXiv* **2019**, arXiv:1904.07094.
47. Reimers, N.; Schiller, B.; Beck, T.; Daxenberger, J.; Stab, C.; Gurevych, I. Classification and Clustering of Arguments with Contextualized Word Embeddings. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Florence, Italy, 2019; pp. 567–578. [CrossRef]
48. Gutiérrez, L.; Keith, B. A Systematic Literature Review on Word Embeddings. In *International Conference on Software Process Improvement*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 132–141.
49. Xin, Y.; Hart, E.; Mahajan, V.; Ruvini, J. Learning Better Internal Structure of Words for Sequence Labeling. *arXiv* **2018**, arXiv:1810.12443.
50. Yuan, H.; Yang, Z.; Chen, X.; Li, Y.; Liu, W. URL2Vec: URL Modeling with Character Embeddings for Fast and Accurate Phishing Website Detection. In Proceedings of the 2018 IEEE International Conference on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), Melbourne, Australia, 11–13 December 2018; pp. 265–272. [CrossRef]

51. Zhai, Z.; Nguyen, D.Q.; Verspoor, K. Comparing CNN and LSTM character-level embeddings in BiLSTM-CRF models for chemical and disease named entity recognition. *arXiv* **2018**, arXiv:1808.08450.
52. Tjong Kim Sang, E.F.; De Meulder, F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, Edmonton, Canada, 31 May–1 June 2003; pp. 142–147.
53. Zhu, S.; Yu, K. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 5675–5679.
54. Pahuja, V.; Laha, A.; Mirkin, S.; Raykar, V.; Kotlerman, L.; Lev, G. Joint learning of correlated sequence labelling tasks using bidirectional recurrent neural networks. *arXiv* **2017**, arXiv:1703.04650.
55. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Machine Learning-Based Code Auto-Completion Implementation for Firmware Developers

Junghyun Kim [1], Kyuman Lee [2,*] and Sanghyun Choi [3]

[1] School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA; andy.kim@gatech.edu

[2] Department of Robot and Smart System Engineering, Kyungpook National University, Daegu 41566, Korea

[3] Memory S/W Development Team, Samsung Electronics, Hwasung 18448, Korea; sh518.choi@samsung.com

* Correspondence: klee400@knu.ac.kr

Received: 29 October 2020; Accepted: 19 November 2020; Published: 28 November 2020

Abstract: With the advent of artificial intelligence, the research paradigm in natural language processing has been transitioned from statistical methods to machine learning-based approaches. One application is to develop a deep learning-based language model that helps software engineers write code faster. Although there have already been many attempts to develop code auto-completion functionality from different research groups, a need to establish an in-house code has been identified for the following reasons: (1) a security-sensitive company (e.g., Samsung Electronics) may not want to utilize commercial tools given that there is a risk of leaked source codes and (2) commercial tools may not be applicable to the specific domain (e.g., SSD firmware development) especially if one needs to predict unique code patterns and style. This research proposes a hybrid approach that harnesses the synergy between machine learning techniques and advanced design methods aiming to develop a code auto-completion framework that helps firmware developers write code in a more efficient manner. The sensitivity analysis results show that the deterministic design results in reducing prediction accuracy as it generates output in some unexpected ways, while the probabilistic design provides a list of reasonable next code elements in which one could select it manually to increase prediction accuracy.

Keywords: machine learning; code auto-completion; GPT-2 model; advanced design methods

1. Introduction

1.1. Research Motivation

Firmware software developers at a company are typically responsible for developing a software program that operates a product. A company always seeks to provide a streamlined work process for firmware software developers to increase productivity as the process leads to saving money for the company. One potential barrier for increasing productivity is to spend considerable time writing code that is particularly due to a repetitive task. Another potential problem is that firmware software developers may be generating similar codes simultaneously as they are separately involved in developing different hardware products. Figure 1 notionally illustrates the issue where we noticed that two firmware software developers separately worked on each code that was eventually similar to each other. This situation could prevent them from working efficiently if they would need to handle a large volume of source codes, resulting in decreasing productivity. Thus, a need to develop a framework that helps firmware software developers work efficiently has been identified in this research.

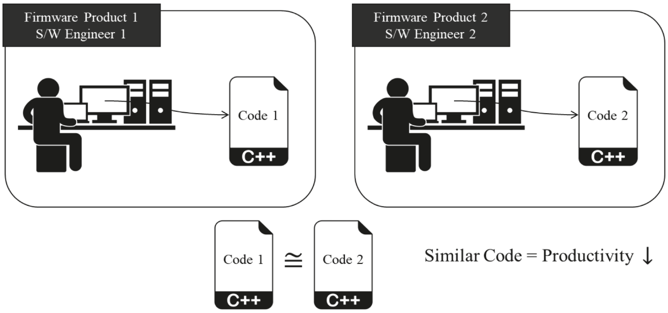


Figure 1. Notional sketch of necessity for developing a code auto-completion framework.

1.2. Background

With the advent of machine learning (ML) techniques, the research paradigm in various engineering areas has been recently transitioned from theory-based to data-driven approaches [1]. There have already been many studies asserting that ML techniques outperform traditional statistical methods. A language model is no exception to this paradigm shift. Many research groups have been committed to developing a language model using ML techniques. For example, Google developed the bidirectional encoder representations from transformers (BERT) [2] that mainly use transformer encoder blocks. OpenAI released the generative pre-trained transformer (GPT) models [3,4] such as the GPT-2 models.

The GPT models are transformer-based language models trained on a massive text dataset from the websites. Depending on the size of neural network weight parameters, the GPT-2 models are classified into small (i.e., 117 M), medium (i.e., 345 M), and large (i.e., 762 M) pre-trained models, as shown in Figure 2. Here, 117 M means that there are 117 million parameters of the neural network model. The obvious upside of the GPT models is that the pre-trained models can be easily tailored to various domain-specific language modeling tasks, given that the models are fine-tuned with domain-specific training datasets. For this reason, the GPT models have been widely used for a variety of domain-specific tasks such as speech recognition and language translation.

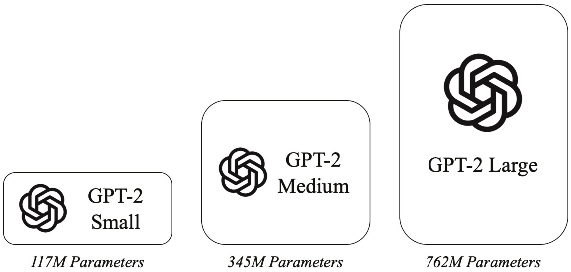
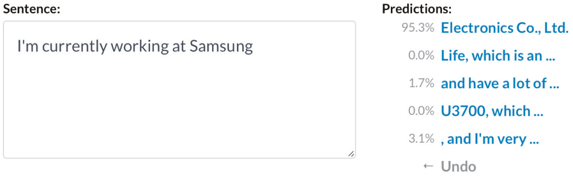


Figure 2. GPT-2 models (reproduced from [5]).

In fact, the GPT models have stunned the world by demonstrating the impressive capability that may exceed the current language models. One example is the Allen AI GPT-2 Explorer [6] as shown in Figure 3 , where it uses the GPT-2 345M model to predict the most likely next word alongside their probability score. In this example, it appears that the model generates a list of candidates for the new few words (e.g., Electronics Co., Ltd.) once some initial text (e.g., I am currently working at Samsung) is provided.



Sentence:

I'm currently working at Samsung

Predictions:

95.3% Electronics Co., Ltd.

0.0% Life, which is an ...

1.7% and have a lot of ...

0.0% U3700, which ...

3.1% , and I'm very ...

← Undo

Figure 3. AllenNLP language modeling demonstration example.

Given the aforementioned observations, it can be hypothesized if the pre-trained GPT-2 models are fine-tuned with SSD firmware source codes, the fine-tuned model predicts the most likely next code element. To that end, this research aims to develop a framework that deploys the GPT-2 models to help SSD firmware developers write code in a more efficient manner. The remainder of this paper consists of the following: Literature Review, Proposed Methodology, Results and Discussion, and Conclusion.

2. Literature Review

In relation to the research objective, there have already been many attempts to develop similar capabilities. This section is aimed at reviewing the advances and limitations of the previous efforts about code auto-completion functionality, which helps identify research gaps that need to be bridged.

2.1. Related Work

Code auto-completion functionality has been considered as one of the most essential functions for software engineers. The Integrated Development Environment (IDE) has provided a set of effective features that include code auto-completion capability [7]. The code auto-completion feature in the IDEs basically suggests next probable code elements; however, there are some potential issues that have been identified [8]: (1) the feature requires an exhaustive set of rules, (2) predictions do not consider the category of code, (3) predictions do not consider context such as class definition, and (4) recommendations are often lexicographical and alphabetical, which may not be very useful.

Many research groups have adopted statistical methods to resolve the potential issues of the IDEs. For example, Sebastian Proksch et al. [9] replaced an existing code auto-completion engine by an approach using Bayesian networks named pattern-based Bayesian network (PBN). Raychev et al. [10] proposed the state-of-the-art probabilistic model for code auto-completion functionality, which is mainly equipped with the n-gram model that computes the probability of the next code elements given previous n elements. The statistical approach, however, examines only a limited number of elements in the source codes when completing the code; thus, the effectiveness of this approach may not scale well to large programs [11].

With the advent of deep learning, many research groups have been committed to developing deep learning-based code auto-completion functionality. The most common technique is to use a recurrent neural network (RNN). In fact, Karampatsis et al. [12] showed that the RNN-based language models would be much better than the traditional statistical methods. Moreover, Martin White et al. [13] illustrated how to use the RNN-based language model to facilitate the code auto-completion task. It seemed that RNN-based language models gained the most popularity at the time; however, it was identified that the models were limited by the so-called hidden state bottleneck: all the information about the current sequence is compressed into a fixed-size vector. This limitation made it hard for the RNN-based models to handle long-range dependencies [14].

A transformer-based language model has been introduced to overcome a major drawback of an RNN-based language model by relying on the attention mechanism. For example, Alexey Svyatkovskiy introduced IntelliCode Compose [15], which is capable of predicting sequences of code tokens of arbitrary types. It leveraged the state-of-the-art generative transformer model trained on 1.2 billion lines of source codes. In addition to the IntelliCode Compose, a variety of transformer-based language models have recently achieved excellent work [2–4,16] for various natural language processing

(NLP) tasks such as language modeling. There are numerous practical applications that deploy a transformer-based language model for code auto-completion functionality. TabNine [17] published a blog post mentioning the use of GPT-2 model in their code auto-completion feature. However, they never revealed technical details about the modeling process. Kite-Pro [18], which also employs the transformer-based language model, reports on average 18 percent more efficiency by using the code auto-completion feature. Table 1 summarizes four different approaches with the advantages and limitations of the previous efforts about code-completion functionality.

Table 1. Comparative table of the related works.

Approach	Example Method	Advantage	Limitation
Integrated Development Environment (IDE)	Eclipse's content assist feature	It provides a list of type-compatible names for the next tokens immediately	It is generally organized alphabetically, which is not very effective
Statistical Language Models	n-gram	It solves a drawback of IDE's ineffectiveness by optimizing/ranking the list	It is difficult to be scaled to large programs
Machine learning-based Language Models	Recurrent Neural Network (RNN)	It is typically better than statistical language models in predictions	It is not effective to capture long-term dependencies
Transformer-based Language Models	Generative Pre-trained Transformer (GPT)	It overcomes the issue related to long-term dependencies by introducing attention mechanisms	It is computationally expensive for those who use a personal computer

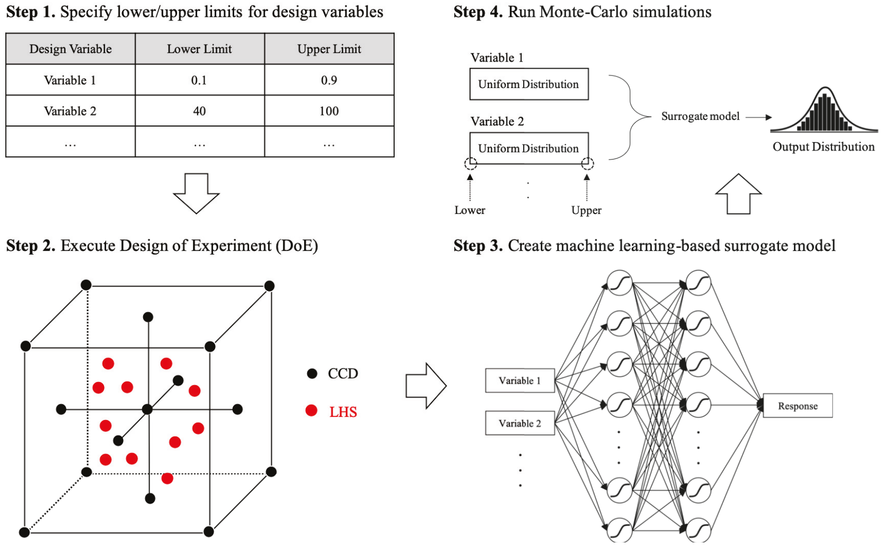
2.2. Research Gap

Although many research groups have deployed a transformer-based language model for code auto-completion functionality, it is important to note that they have trained the model, with open source codes mostly coming from GitHub. For example, the Deep TabNine [17] is trained on around 2 million files from GitHub. This indicates that the software may not be applicable if one needs to predict very unique code patterns and style. Therefore, a need to establish a domain-specific language model has been identified based on the following reasons: (1) firmware codes implement very specific sets of features for the hardware and (2) firmware codes typically comply with unique coding styles and patterns optimized for embedded environments.

In fact, some companies (e.g., TabNine) advertise that they offer a GPU-based cloud service that enables users to create a custom model by fine-tuning the model with their input data. However, a security-sensitive company such as Samsung Electronics may not want to utilize the commercial tools given that there is a risk of leaked source codes. In addition to the security issue, a company has to pay for the license fee because the tools are not free to use the service. Thus, a need to develop in-house codes for code auto-completion functionality is identified.

As we seek to develop a domain-specific (i.e., solid state drive (SSD) firmware development) language model by using the GPT-2 model, it naturally leads us to consider how to determine diversity parameters (i.e., *Top_k*, *Top_p*, and Boltzmann temperature) of the model. Given that there has not been any analysis done on the optimal diversity parameter values especially on the SSD firmware development domain, the following research question can be constructed: "How can we determine the GPT-2 diversity parameter values properly for the SSD firmware development domain"?

To answer the question, in this paper, we propose a hybrid approach that harnesses the synergy between ML techniques and advanced design methods (e.g., design of experiment, surrogate modeling, and Monte Carlo simulation) [19] to enhance the level of understanding of the relationship between the GPT-2 model diversity parameters and code auto-completion functionality in the SSD firmware development domain. Figure 4 notionally illustrates the process of the hybrid approach used for this research.


Step 1. Specify lower/upper limits for design variables
Step 4. Run Monte-Carlo simulations
Variable 1
Uniform Distribution
Variable 2
Uniform Distribution
Surrogate model
Output Distribution
Lower
Upper
Step 2. Execute Design of Experiment (DoE)
Step 3. Create machine learning-based surrogate model
CCD
LHS
Variable 1
Variable 2
Response

Design Variable	Lower Limit	Upper Limit
Variable 1	0.1	0.9
Variable 2	40	100
…	…	…

Figure 4. Notional sketch of the process of the hybrid approach used for this research.

3. Proposed Methodology

3.1. Overview of the Methodology

This research aims to not only develop a framework that deploys GPT-2 117M model to help firmware developers write code in a more efficient manner, but also unravel the hidden relationships between the GPT-2 model diversity parameters and code auto-completion capability. Figure 5 depicts an overview of the proposed methodology.

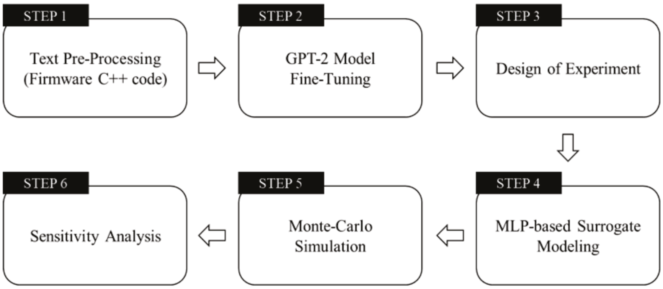


Figure 5. Overview of the proposed methodology.

The framework is a Python-based program that consists of several modules with its primary data sources. There are three different modules: (1) the first module is designed to automate data pre-processing, such as removing all unnecessary C++ code comments, (2) the second module performs fine-tuning for the GPT-2 model with optimized hyper-parameters (i.e., batch size and learning rate), and (3) the third module employs advanced design methods for diversity parameter sensitivity analysis.

3.2. Text Pre-Processing

To customize the original GPT-2 117M model to the SSD firmware development domain, it is imperative to prepare input data properly for the fine-tuning process, because the process may require understanding input data in its own way. Since the source codes include unnecessary information (e.g., C++ code comments) that may deteriorate training data quality, we develop a Python code that automatically removes all unnecessary code through the pattern analysis. Once the Python code completes the removal process, it also removes white spaces as well as empty lines. It then combines all the source codes into one single text file with the delimiter in order to allow the model to learn the formatting of the training data.

3.3. GPT-2 Model Fine-Tuning

The GPT-2 model is a transformer-based language model trained with a massive 40GB text data that mainly includes web pages [18]. Users can fine-tune the GPT-2 model with new input data. During the fine-tuning process, users can either increase or decrease two hyper-parameters, namely batch size and learning rate, to optimize the model's predictive capability. We employ the grid search method as shown in Figure 6 and test all candidate cases on the NVIDIA DGX-1 machine (i.e., Volta 32GB version) to isolate the hyper-parameters. The effective model is finally determined by minimizing the log-loss value. The choice of hyper-parameters is tabulated in Table 2. Figure 7 shows the plot of the loss curve describing that the loss value is actually converged with the chosen hyper-parameters.

Table 2. Grid search results for the hyper-parameters.

Hyper-Parameter	Final Choice
Learning rate	0.0001
Batch size	Stochastic

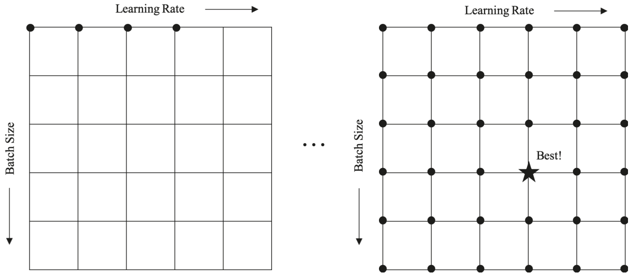


Figure 6. Notional sketch of the grid search method for isolating hyper-parameters.

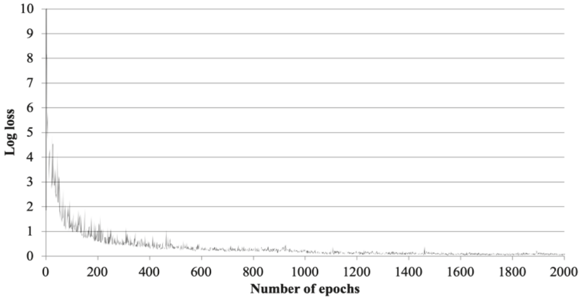


Figure 7. Plot of the loss history of the tailored GPT-2 model with isolated parameters.

3.4. Design of Experiment

The design of experiment (DoE) is a procedure that selects samples in the design space to maximize the amount of information with a limited set of experiments. To generate a non-linear surrogate model that represents the design space of the GPT-2 model's diversity parameters, we employ two representative DoE methods: (1) the Latin hypercube sampling (LHS) method is used to capture inner points of the design space and (2) the full factorial design with three factors is utilized to capture corner points of the design space. Figure 8 shows how samples are distributed in the design space of the GPT-2 model's diversity parameters.

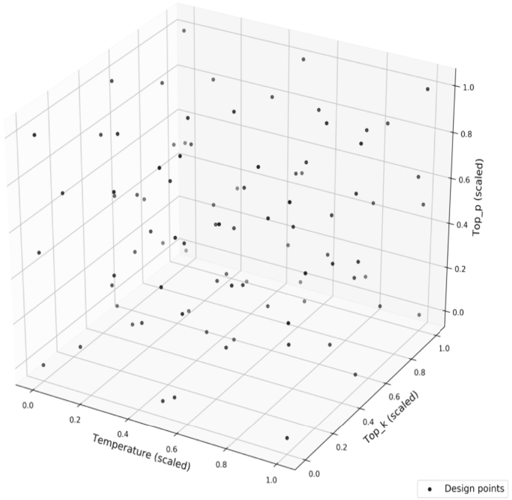


Figure 8. Design space of the GPT-2 model's diversity parameters.

3.5. Surrogate Modeling

The multi-layer perceptron (MLP), which is one of the most representative non-linear regression methods, is deployed as a surrogate model with respect to the GPT-2 model's diversity parameters. The MLP model used for this research entails the following fully-connected layers: (1) an input layer to receive diversity parameter values, (2) an output layer that makes a prediction in terms of the score function illustrated in Table 3, and (3) two hidden layers that are the true computational engine for the regression. Figure 9 shows a diagram of the MLP model structure used for this research.

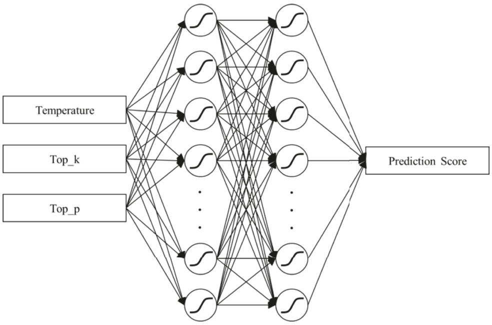


Figure 9. Diagram of the MLP model structure.

Table 3. Score metric for the MLP-based surrogate modeling.

Case	Score
An option with the highest probability matches the actual code	100
There is an option among the possible candidates, which matches to the actual code	50
There is no available option that matches the actual code	1

To evaluate the accuracy of the MLP-based surrogate model, the model representation error (MRE) is calculated with respect to additional random DoE cases. As a result, R-square, which describes how well the model predictions adhere to reality, is equal to 0.98 and root mean square error (RMSE), which describes how to spread out the residuals, is approximately 3.12.

3.6. Monte Carlo Simulation

We utilize the Monte Carlo simulation (MCS) technique to see the trend of resulting outcomes generated from the MLP-based surrogate model. Uniform distribution with min/max values is used for the GPT-2 model's diversity parameters. The MCS is then performed with 1,000,000 sample points generated by the uniform distribution of the GPT-2 model's diversity parameters, which are eventually incorporated into the MLP-based surrogate model to yield statistical distributions. Figure 10 notionally depicts the MCS process flow diagram especially used in this research (i.e., input and output mapping).

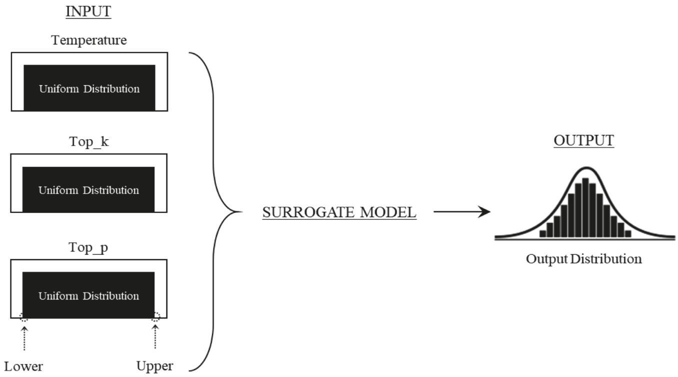


Figure 10. Notional sketch of the MCS process flow diagram.

4. Results and Discussion

4.1. Sensitivity Analysis

Sensitivity analysis with respect to the GPT-2 diversity parameters is performed to enhance the level of understanding of the relationship between prediction accuracy and the diversity parameters. The GPT-2 model has three different diversity parameters implemented in the sampling process.

The Boltzmann temperature is one of the GPT-2 model diversity parameters that control randomness in the sampling process. Figure 11 shows the MCS results with respect to the Boltzmann temperature. Lower and upper bounds are specified with 0.1 and 0.9, respectively. A black dot represents one experiment case generated by the MLP-based surrogate model with three different input variables randomly sampled from the uniform distributions with respect to the GPT-2 model's diversity parameters. As can be seen from Figure 11, it seems that decreasing Boltzmann temperature (i.e., *x*-axis) keeps the model to generate a high prediction score (i.e., *y*-axis), while increasing Boltzmann temperature causes the model to tend to frequently have a low prediction score. Based on these observations, one may claim that the model with lower Boltzmann temperature value, named deterministic design in this paper, would be the best option to predict the most likely next code element as the deterministic design strives to minimize the degree of surprise in model output. However, it is too early to draw such a conclusion, because the model with a higher Boltzmann temperature value, named probabilistic design in this paper, would provide a list of reasonable next code elements in which one could select it manually to increase prediction accuracy. Details will be discussed in the section of model evaluation.

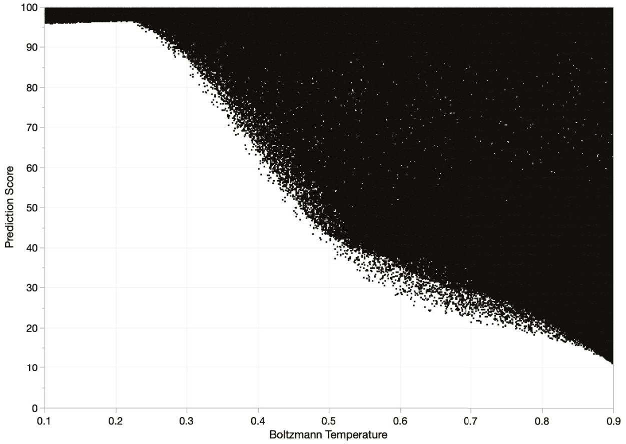


Figure 11. MCS results for diversity parameter 1 (i.e., Boltzmann temperature).

The *Top_k* is another GPT-2 model diversity parameter that controls the number of sampling words to be considered. For example, the most likely word is only considered if the *Top_k* is equal to onem thus resulting in deterministic design. The deterministic design can successfully eliminate rather weird candidates; however, one may claim that better results would be achieved if the algorithm considers sampling words more than one. Figure 12 shows the MCS results with respect to the *Top_k* parameter. Lower and upper bounds are specified with 40 and 100, respectively. As can be seen, it appears that the *Top_k* parameter value does not have a significant impact on the prediction score, indicating that the *Top_k* parameter may not entirely contribute to the model output diversity.

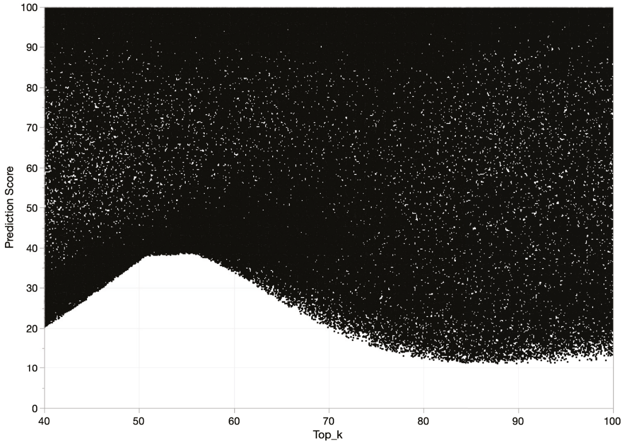


Figure 12. MCS results for diversity parameter 2 (i.e., *Top_k*).

The *Top_p*, one of the GPT-2 model diversity parameters, considers sampling words from the largest possible set of words whose cumulative probability exceeds a user-defined number. Instead of sampling only from the most likely *K* words, the *Top_p* parameter provides an option that dynamically controls the size of the set of sampling words to be considered. For example, if the *Top_p* is equal to 0.9, the algorithm computes cumulative probability distribution (CDF) and cuts off the words as soon as the CDF exceeds 90 percents. Figure 13 shows the MCS results with respect to the *Top_p* parameter. Lower and upper bounds are specified with 0.1 and 0.9, respectively. As the *Top_p* parameter value increases, it results in more randomness in terms of the prediction score. On the other hand, as the *Top_p* parameter value decreases, it leads to less randomness with regard to the prediction score. This implies that the model with lower *Top_p* parameter value, which is approximately 0.25 in this case, becomes deterministic and repetitive; while, the model with a higher *Top_p* parameter value becomes a probabilistic design that may relatively improve code suggestion quality compared to a deterministic design. Details about the difference between deterministic and probabilistic design will be discussed in the section of model evaluation.

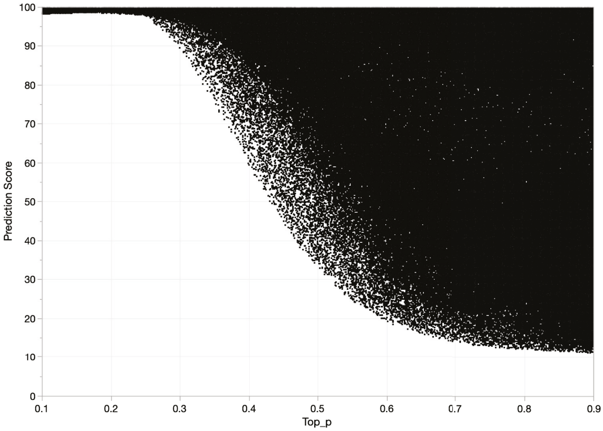
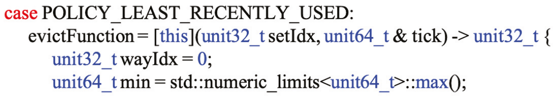


Figure 13. MCS results for diversity parameter 3 (i.e., *Top_p*).

4.2. Model Evaluation

The simplest way to evaluate the fine-tuned GPT-2 model is to allow the model to ramble on its own, which is called generating unconditional samples, but we are determined to use the option, called generating interactive conditional samples, for a model evaluation purpose, as it is easy to steer customized samples. The interactive conditional sample refers to generating samples based on a user-defined input code. For example, it generates the most likely next code element once the user provides an initial code. After users select the code element, the element is then added to the sequence of the input code. Then, a new sequence becomes the input for the next step. This process is repeated until it fills the rest of the sequence. In this paper, we use open-source SSD firmware codes, namely SimpleSSD [20], to evaluate the framework developed by this research, because Samsung Electronics SSD firmware source codes are strictly confidential. Figure 14 shows one sample code element [21] tested by the framework.



```
case POLICY_LEAST_RECENTLY_USED:
  evictFunction = [this](unit32_t setIdx, unit64_t & tick) -> unit32_t {
    unit32_t wayIdx = 0;
    unit64_t min = std::numeric_limits<unit64_t>::max();
```

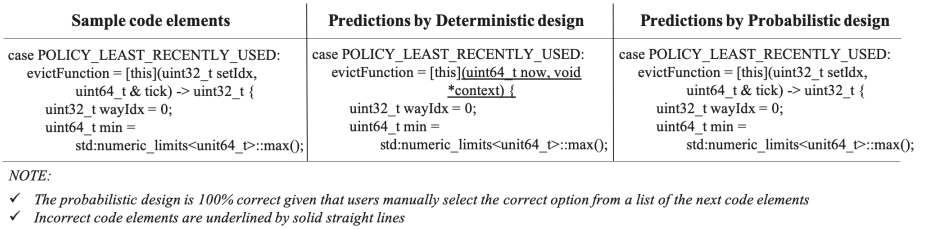
Figure 14. Sample code element to be tested.

Based on the sensitivity analysis results, we specify the parameter values tabulated in Table 4 for the deterministic design. It should be noted that we specify 40 for the *Top_k* parameter (i.e., rule of thumb) [22], as the parameter does not affect randomness in the sampling process. Regarding the probabilistic design, we specify the maximum value (i.e., upper limit) for the GPT-2 diversity parameters except for the *Top_k* parameter.

Table 4. Diversity parameter values for deterministic design.

Diversity Parameter	Value
Boltzmann temperature	0.22
Top_k	40
Top_p	0.25

Figure 15 shows the results of predictions by the deterministic and probabilistic design for the sample code element from Figure 14. Incorrect code elements are underlined by solid straight lines. This result indicates that the probabilistic design is better than the deterministic design with respect to similarity, especially for the sample code element. Here, it must be noted that the probabilistic design is 100% correct as the users could select the correct code element after the framework suggest a list of the most likely next code elements. Furthermore, it is worth mentioning that the deterministic design is capable of predicting the correct next code elements in most cases; however, it sometimes produces the model output in some unexpected ways.



Sample code elements	Predictions by Deterministic design	Predictions by Probabilistic design

case POLICY_LEAST_RECENTLY_USED:
evictFunction = [this](uint32_t setIdx,
uint64_t & tick) -> uint32_t {
uint32_t wayIdx = 0;
uint64_t min =
std:numeric_limits<unit64_t>::max();
	case POLICY_LEAST_RECENTLY_USED:
	evictFunction = [this](uint64_t now, void
	*context) {
	uint32_t wayIdx = 0;
	uint64_t min =
	std:numeric_limits<unit64_t>::max();
		case POLICY_LEAST_RECENTLY_USED:
		evictFunction = [this](uint32_t setIdx,
		uint64_t & tick) -> uint32_t {
		uint32_t wayIdx = 0;
		uint64_t min =
		std:numeric_limits<unit64_t>::max();

NOTE:

✓ *The probabilistic design is 100% correct given that users manually select the correct option from a list of the next code elements*

✓ *Incorrect code elements are underlined by solid straight lines*

Figure 15. Sample code element predictions by deterministic and probabilistic design.

Figure 16 shows how deterministic and probabilistic design predicts the next code element differently based on the same user-defined input. As can be seen, the deterministic design generated the only one output that would be the most likely (i.e., 100% probability) next code element in the prediction process. Unfortunately, this prediction was incorrect for this particular case even though the deterministic design was typically able to predict correctly in most cases. In other words, the reason why the deterministic design would not be able to predict the code element correctly for this case was that it would automatically select the option with the highest prediction probability. On the other hand, given the same input code element, the probabilistic design generated a list of the most likely next code elements with a certain probability. This approach enabled the users to select the option manually from the list that included the correct code element based on previous input. For example, although the option with the highest probability (e.g., [this](unit_64t) was incorrect for this particular case, the probabilistic design was able to increase prediction accuracy by allowing the users to select an option (e.g., [this](unit_32t) manually. Thus, there was no discrepancy with the probabilistic design given that the user could manually select the candidate from the list of options (e.g., choosing an option with 20% probability instead of an option with 80% probability).

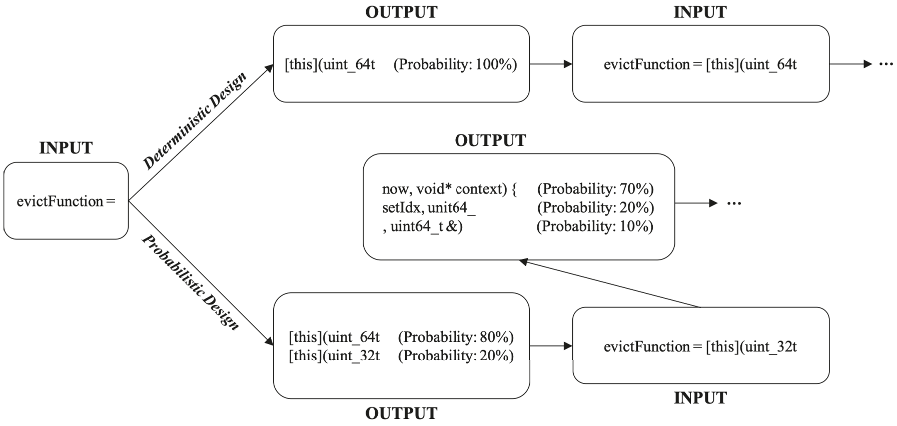
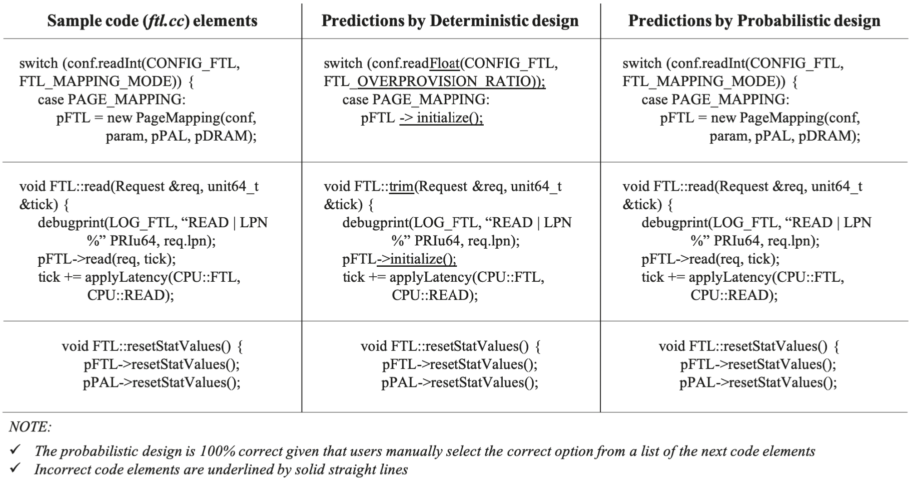

OUTPUT
INPUT
[this](uint_64t (Probability: 100%)
evictFunction = [this](uint_64t
···
Deterministic Design
INPUT
evictFunction =
OUTPUT
now, void* context) { (Probability: 70%)
setIdx, unit64_ (Probability: 20%)
, uint64_t &) (Probability: 10%)
···
Probabilistic Design
[this](uint_64t (Probability: 80%)
[this](uint_32t (Probability: 20%)
evictFunction = [this](uint_32t
INPUT
OUTPUT

Figure 16. Deterministic design vs. Probabilistic design.

This example shows the impact of the GPT-2 model diversity parameters on the model output prediction accuracy. In addition to this particular example, Figures 17–19 support the argument that the deterministic design sometimes generates the model output in some unexpected ways, while there is no discrepancy with the probabilistic design, given that the user could manually select an option from a list of candidates.



Sample code (*ftl.cc*) elements	Predictions by Deterministic design	Predictions by Probabilistic design

switch (conf.readInt(CONFIG_FTL,
FTL_MAPPING_MODE)) {
case PAGE_MAPPING:
pFTL = new PageMapping(conf,
param, pPAL, pDRAM);
	switch (conf.readFloat(CONFIG_FTL,
	FTL_OVERPROVISION_RATIO));
	case PAGE_MAPPING:
	pFTL -> initialize();
		switch (conf.readInt(CONFIG_FTL,
		FTL_MAPPING_MODE)) {
		case PAGE_MAPPING:
		pFTL = new PageMapping(conf,
		param, pPAL, pDRAM);

void FTL::read(Request &req, unit64_t
&tick) {
debugprint(LOG_FTL, "READ | LPN
%" PRIu64, req.lpn);
pFTL->read(req, tick);
tick += applyLatency(CPU::FTL,
CPU::READ);
	void FTL::trim(Request &req, unit64_t
	&tick) {
	debugprint(LOG_FTL, "READ | LPN
	%" PRIu64, req.lpn);
	pFTL->initialize();
	tick += applyLatency(CPU::FTL,
	CPU::READ);
		void FTL::read(Request &req, unit64_t
		&tick) {
		debugprint(LOG_FTL, "READ | LPN
		%" PRIu64, req.lpn);
		pFTL->read(req, tick);
		tick += applyLatency(CPU::FTL,
		CPU::READ);

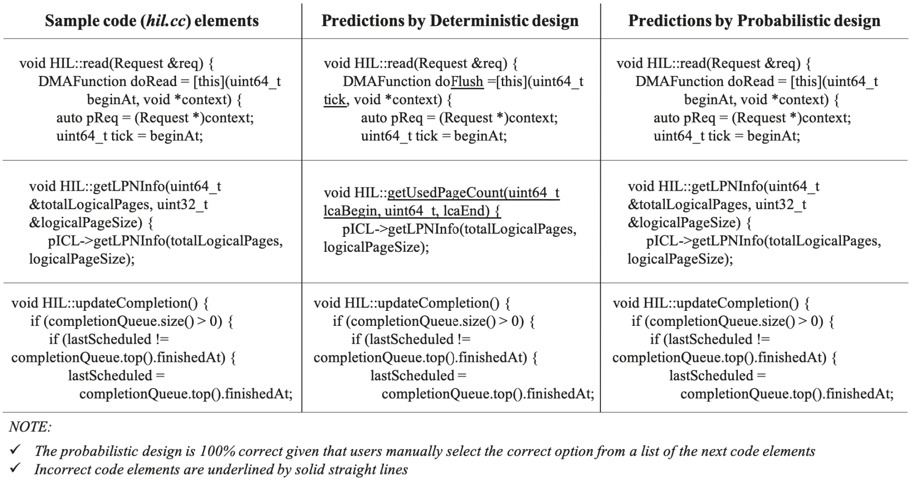
void FTL::resetStatValues() {
pFTL->resetStatValues();
pPAL->resetStatValues();
	void FTL::resetStatValues() {
	pFTL->resetStatValues();
	pPAL->resetStatValues();
		void FTL::resetStatValues() {
		pFTL->resetStatValues();
		pPAL->resetStatValues();

NOTE:

- ✓ *The probabilistic design is 100% correct given that users manually select the correct option from a list of the next code elements*
- ✓ *Incorrect code elements are underlined by solid straight lines*

Figure 17. Sample code (*ftl.cc*) element predictions by deterministic and probabilistic design.

The main gist of these case studies is as follows: First, the deterministic design is recommended for those who would like to reduce latency time for the model output prediction, but users should recognize that the prediction accuracy may not be guaranteed. Second, the probabilistic design is recommended for those who want to guarantee the model prediction accuracy; however, users may have to address potential issues related to high computational costs about inference. Regarding the potential issues, we recognize the trend that transformer-based language models are going to get bigger (e.g., GPT-3 model), so they may require a lot of computing power and memory to run them, which potentially leads to challenging to run on a personal computer with reasonable latency time. In this case, it is imperative to implement the code auto-completion engine developed by this research into a cloud computing resource. The good news is that Samsung Electronics already has an environment internally that includes many cloud computing systems. With the internal systems, the company does not have to account for security concerns. However, one potential issue is that the cloud systems may experience a bottleneck because of an increase in traffic from user requests. This paper does not address the potential limitation about how to operate the framework developed by this research with the cloud computing systems, but focus on proving the concept of the machine learning-based code auto-completion functionality.



Sample code (*hil.cc*) elements	Predictions by Deterministic design	Predictions by Probabilistic design

void HIL::read(Request &req) {
DMAFunction doRead = [this](uint64_t
beginAt, void *context) {
auto pReq = (Request *)context;
uint64_t tick = beginAt;
	void HIL::read(Request &req) {
	DMAFunction doFlush =[this](uint64_t
	tick, void *context) {
	auto pReq = (Request *)context;
	uint64_t tick = beginAt;
		void HIL::read(Request &req) {
		DMAFunction doRead = [this](uint64_t
		beginAt, void *context) {
		auto pReq = (Request *)context;
		uint64_t tick = beginAt;

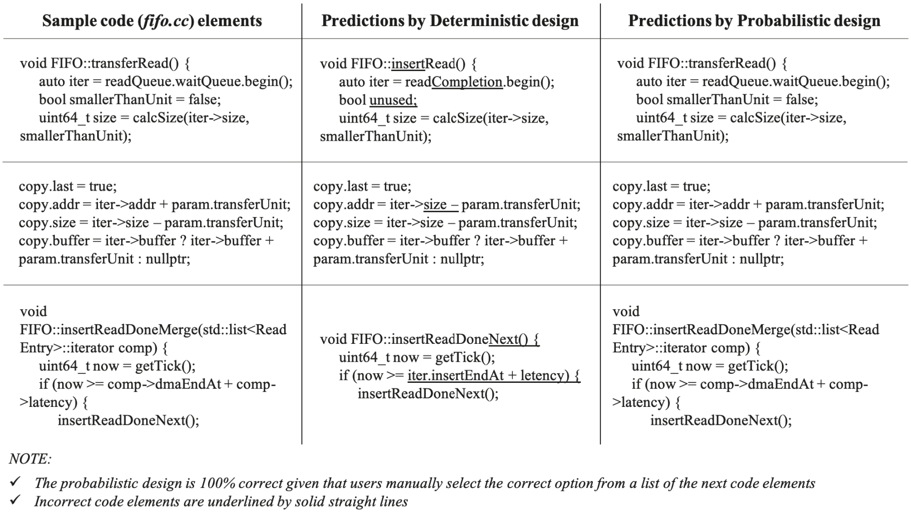
void HIL::getLPNInfo(uint64_t
&totalLogicalPages, uint32_t
&logicalPageSize) {
pICL->getLPNInfo(totalLogicalPages,
logicalPageSize);
	void HIL::getUsedPageCount(uint64_t
	lcaBegin, uint64_t, lcaEnd) {
	pICL->getLPNInfo(totalLogicalPages,
	logicalPageSize);
		void HIL::getLPNInfo(uint64_t
		&totalLogicalPages, uint32_t
		&logicalPageSize) {
		pICL->getLPNInfo(totalLogicalPages,
		logicalPageSize);

void HIL::updateCompletion() {
if (completionQueue.size() > 0) {
if (lastScheduled !=
completionQueue.top().finishedAt) {
lastScheduled =
completionQueue.top().finishedAt;
	void HIL::updateCompletion() {
	if (completionQueue.size() > 0) {
	if (lastScheduled !=
	completionQueue.top().finishedAt) {
	lastScheduled =
	completionQueue.top().finishedAt;
		void HIL::updateCompletion() {
		if (completionQueue.size() > 0) {
		if (lastScheduled !=
		completionQueue.top().finishedAt) {
		lastScheduled =
		completionQueue.top().finishedAt;

NOTE:

- ✓ *The probabilistic design is 100% correct given that users manually select the correct option from a list of the next code elements*
- ✓ *Incorrect code elements are underlined by solid straight lines*

Figure 18. Sample code (*hil.cc*) element predictions by deterministic and probabilistic design.



Sample code (*fifo.cc*) elements	Predictions by Deterministic design	Predictions by Probabilistic design

void FIFO::transferRead() {
auto iter = readQueue.waitQueue.begin();
bool smallerThanUnit = false;
uint64_t size = calcSize(iter->size,
smallerThanUnit);
	void FIFO::insertRead() {
	auto iter = readCompletion.begin();
	bool unused;
	uint64_t size = calcSize(iter->size,
	smallerThanUnit);
		void FIFO::transferRead() {
		auto iter = readQueue.waitQueue.begin();
		bool smallerThanUnit = false;
		uint64_t size = calcSize(iter->size,
		smallerThanUnit);

copy.last = true;
copy.addr = iter->addr + param.transferUnit;
copy.size = iter->size – param.transferUnit;
copy.buffer = iter->buffer ? iter->buffer +
param.transferUnit : nullptr;
	copy.last = true;
	copy.addr = iter->size – param.transferUnit;
	copy.size = iter->size – param.transferUnit;
	copy.buffer = iter->buffer ? iter->buffer +
	param.transferUnit : nullptr;
		copy.last = true;
		copy.addr = iter->addr + param.transferUnit;
		copy.size = iter->size – param.transferUnit;
		copy.buffer = iter->buffer ? iter->buffer +
		param.transferUnit : nullptr;

void
FIFO::insertReadDoneMerge(std::list<Read
Entry>::iterator comp) {
uint64_t now = getTick();
if (now >= comp->dmaEndAt + comp-
>latency) {
insertReadDoneNext();
	void FIFO::insertReadDoneNext() {
	uint64_t now = getTick();
	if (now >= iter.insertEndAt + letency) {
	insertReadDoneNext();
		void
		FIFO::insertReadDoneMerge(std::list<Read
		Entry>::iterator comp) {
		uint64_t now = getTick();
		if (now >= comp->dmaEndAt + comp-
		>latency) {
		insertReadDoneNext();

NOTE:

- ✓ *The probabilistic design is 100% correct given that users manually select the correct option from a list of the next code elements*
- ✓ *Incorrect code elements are underlined by solid straight lines*

Figure 19. Sample code (*fifo.cc*) element predictions by deterministic and probabilistic design.

5. Conclusions

In this research, we established a machine learning-based code auto-completion framework, especially for SSD firmware developers at Samsung Electronics. The hybrid approach that harnesses the synergy between machine learning techniques and advanced design methods was presented to enhance the level of the understanding of the relationship between the GPT-2 model diversity parameters and prediction accuracy. The sensitivity analysis results showed that the probabilistic design outperformed the deterministic design with respect to the prediction accuracy as we observed a few cases of failure showing that the deterministic design generated output in some unexpected ways. It was found that there must be a balance between model prediction accuracy and prediction latency time given that users utilize the framework with either laptops or desktops. The accomplishment of this research can be implemented in any firmware development environment at a company as needed. In conclusion, it is expected that the framework developed by this research can save numerous hours of productivity by eliminating tedious parts of writing code and helping SSD firmware developers write code in a more efficient manner. Future research will extend the framework by implementing a new functionality accounting for potential issues related to the order of suggestions given that users may select accidentally the first entry (i.e., unwanted selections), which may not always be the correct option, among the recommended code elements.

Author Contributions: Conceptualization, J.K. and S.C.; Methodology, J.K.; Software, J.K. and S.C.; Validation, J.K.; Investigation, J.K. and K.L.; Writing—original draft preparation, J.K.; Writing—review and editing, K.L. and S.C.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Kyungpook National University Research Fund, 2020.

Acknowledgments: This paper is an extension of a PhD summer internship project that was done at Samsung Electronics in 2020. We would like to thank Hankyu Lee for his feedback on this research. We would also like to thank Jinbaek Song for his support on the NVIDIA DGX-1 setup process.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kim, J.; Briceno, S.; Justin, C.; Mavris, D. A Data-Driven Approach Using Machine Learning to Enable Real-Time Flight Path Planning. In Proceedings of the AIAA Aviation 2020 Forum, USA, 10 November 2020; p. 2873. Available online: https://arc.aiaa.org/doi/abs/10.2514/6.2020-2873 (accessed on 12 November 2020).
2. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
3. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
4. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
5. The illustrated GPT-2 (Visualizing Transformer Language Models. Available online: http://jalammar.github.io/illustrated-gpt2/ (accessed on 12 November 2020).
6. AllenNLP Language Modeling. Available online: https://demo.allennlp.org/next-token-lm?text=AllenNLP%20is (accessed on 12 November 2020).
7. Working with Content Assist. Available online: https://www.eclipse.org/pdt/help/html/working_with_code_assist.htm (accessed on 19 October 2020).
8. Das, S.; Shah, C. *Contextual Code Completion Using Machine Learning*; Technical Report; Stanford University: Stanford, CA, USA, 2015.
9. Proksch, S.; Lerch, J.; Mezini, M. Intelligent code completion with Bayesian networks. *ACM Trans. Softw. Eng. Methodol.* **2015**, *25*, 1–31.
10. Raychev, V.; Vechev, M.; Yahav, E. Code completion with statistical language models. In Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, Edinburgh, UK, 9–11 June 2014; pp. 419–428.
11. Neural Code Completion. Available online: https://openreview.net/pdf?id=rJbPBt9lg (accessed on 27 November 2020).
12. Karampatsis, R.M.; Babii, H.; Robbes, R.; Sutton, C.; Janes, A. Big Code!= Big Vocabulary: Open-Vocabulary Models for Source Code. *arXiv* **2020**, arXiv:2003.07914.

13. White, M.; Vendome, C.; Linares-Vásquez, M.; Poshyvanyk, D. Toward deep learning software repositories. In Proceedings of the 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, Florence, Italy, 16–17 May 2015; pp. 334–345.
14. Li, J.; Wang, Y.; Lyu, M.R.; King, I. Code completion with neural attention and pointer networks. *arXiv* **2017**, arXiv:1711.09573.
15. Svyatkovskiy, A.; Deng, S.K.; Fu, S.; Sundaresan, N. IntelliCode Compose: Code Generation Using Transformer. *arXiv* **2020**, arXiv:2005.08025.
16. Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; Hon, H.W. Unified language model pre-training for natural language understanding and generation. *arXiv* **2019**, arXiv:1905.03197.
17. Autocompletion with Deep Learning. Available online: https://www.tabnine.com/blog/deep/ (accessed on 19 October 2020).
18. Better Language Models and Their Implications. Available online: https://openai.com/blog/better-language-models/ (accessed on 19 October 2020).
19. Kim, J.; Lim, D.; Monteiro, D.J.; Kirby, M.; Mavris, D. Multi-objective Optimization of Departure Procedures at Gimpo International Airport. *Int. J. Aeronaut. Space Sci.* **2018**, *19*, 534–541. [CrossRef]
20. Jung, M.; Zhang, J.; Abulila, A.; Kwon, M.; Shahidi, N.; Shalf, J.; Kim, N.S.; Kandemir, M. Simplessd: Modeling solid state drives for holistic system simulation. *IEEE Comput. Archit. Lett.* **2017**, *17*, 37–41. [CrossRef]
21. SimpleSSD Version 2.0 GitHub. Available online: https://github.com/SimpleSSD/SimpleSSD (accessed on 19 October 2020).
22. Beginner's Guide to Retrain GPT-2 (117M) to Generate Custom Text Content. Available online: https://medium.com/@ngwaifoong92/beginners-guide-to-retrain-gpt-2-117m-to-generate-custom-text-content-8bb5363d8b7f (accessed on 19 October 2020).

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

An Improvement on Estimated Drifter Tracking through Machine Learning and Evolutionary Search

Yong-Wook Nam [1], Hwi-Yeon Cho [1], Do-Youn Kim [2], Seung-Hyun Moon [1] and Yong-Hyuk Kim [1,*]

[1] Department of Computer Science, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea; yonguk6726@naver.com (Y.-W.N.); hwyncho@kw.ac.kr (H.-Y.C.); uramoon@kw.ac.kr (S.-H.M.)

[2] ARA Consulting & Technology, Ltd., 30, Songdomirae-ro, Yeonsu-gu, Incheon 21990, Korea; kimdoyon@empas.com

* Correspondence: yhdfly@kw.ac.kr

Received: 28 September 2020; Accepted: 13 November 2020; Published: 16 November 2020

Abstract: In this study, we estimated drifter tracking over seawater using machine learning and evolutionary search techniques. The parameters used for the prediction are the hourly position of the drifter, the wind velocity, and the flow velocity of each drifter position. Our prediction model was constructed through cross-validation. Trajectories were affected by wind velocity and flow velocity from the starting points of drifters. Mean absolute error (MAE) and normalized cumulative Lagrangian separation (NCLS) were used to evaluate various prediction models. Radial basis function network showed the lowest MAE of 0.0556, an improvement of 35.20% over the numerical model MOHID. Long short-term memory showed the highest NCLS of 0.8762, an improvement of 6.24% over MOHID.

Keywords: drifter trajectory; evolutionary computation; machine learning; deep learning; NCLS

1. Introduction

The worldwide increase of large ships and maritime transportation volume causes a number of accidents that are beyond the capacity of individual nations. Pollutants released from accidents may extensively contaminate the marine environment due to ocean currents, weather, and weathering. Therefore, pollutants released during an accident should be removed as soon and as much as possible. An accurate prediction of the pollutant movement can help track and address them, as a variety of studies have proven [1–5]. The prediction model for oil spills generally calculates the movement and spread of the oil spill using the Lagrangian particle approach [5–7]. This forecasting method uses physics for critical parameters such as flow velocity, wind velocity, water level, and temperature in the current state. Parameter optimization using evolutionary computation [8] showed better results than MOHID [6], a numerical water modelling system. Machine learning and ensemble methods [9,10] were also used to estimate the movement of drifters.

Predicting the movement of a drifter on the ocean is an essential step in tracking the spread of an oil spill [11]. While spilled oil may sink or evaporate, most of it floats on the surface. Conventional numerical models can predict oil spills more accurately if the trajectories of drifting particles can be predicted accurately. Therefore, we aimed to predict more accurate trajectories by using various machine learning methods including deep learning, which has attracted much recent attention, rather than by using evolutionary computation as in our previous study [8]. We integrated artificial intelligence (AI) technology to predict the future ocean state, utilizing the continuity of parameter data over time. We expanded the area covered by our previous study [8], which predicted the trajectory of drifting objects in the ocean and systematically compared a wide range of regression functions and artificial neural networks for predicting the movement of drifters. In order to avoid the look-ahead bias, we used wind and flow forecasts made by the Korea Meteorological Administration and private technical agencies instead of actual future parameters.

In sum, the contributions of this study are summarized as follows. We applied evolutionary computation and machine learning to the prediction of drifter trajectories. We extended and improved our previous study [8] that used evolutionary computation, and we also first predicted the trajectories using various machine learning techniques. This study was the first time that machine learning techniques have been applied to the prediction of drifter trajectories—before this study, only numerical models such as MOHID have been applied to drifter trajectories. Our methods could significantly improve on the results of the representative numerical model, MOHID.

2. Literature Review

ADIOS [12], developed by the United States in the early 1990s, is one of the widely used decision-making support systems for spilled oil. These support systems share the characteristics of simplicity, high performance, open-source programming, and an extensive oil database. The performance of ADIOS has continued to improve [13] and provides a basis for newly developed oil weathering models. Unfortunately, it cannot use data on the current state [14] and simulate the trajectory of the oil spill. The ADIOS model requires information on spilled oil situations, environmental conditions and prevention strategies, and calculates optimal prediction results by inputting minimum information obtained or expected in the field.

The prediction models for oil spill movements were developed for accurate and detailed analysis. Oil companies, consulting sectors, national agencies, and research centers use certain models worldwide, which enable them to input various marine weather and environmental data to consider oil weathering, and are thus suitable for planning stages and research scenarios for different types of oil and marine weather conditions. Related models are GNOME [7], OILMAP [15] OSCAR [16], OSIS, GULFSPILL [17], MOHID [6], etc. Among them, MOHID [6], which is used as a benchmark measure of performance in this study, was first developed in 1985 at the Marine and Environmental Technology Research Center (IST) of the Instituto Superior Técnico (IST), affiliated with the University of Lisbon, Portugal. MOHID is a multifunctional three-dimensional numerical analysis model that can be applied to coastal and estuary areas, which basically calculates physical actions in coastal and estuary areas such as tide and tsunami. It consists of more than 60 modules that can calculate fluid properties (water temperature, salinity, etc.), Lagrangian movement, turbulence, sediment movement, erosion and sedimentation, meteorological and wave conditions, water quality and ecology, and oil diffusion.

Recent advancements in operational maritime, weather forecasting, and computing technologies have allowed for automated prediction in desktop and web environments. Such prediction models include MOTHY [18], POSEIDON OSM [19], MEDSLICK [20], MEDSLICK II [5], OD3D [21], LEEWAY [22], OILTRANS [14], BSHmod.L [23], and SEATRACK Web [24]. Some models cannot process 3D ocean data or consider stokes drift and the vertical movement of droplets. Research is underway to provide user requirements, convenient and comprehensive user-friendly environments, and geographic information system (GIS) results to improve the model landscape.

Typical models that predict Lagrangian drifter trajectory are known to be less accurate for large structures [25]. Therefore, Aksamit et al. [25] used long short-term memory (LSTM) to accurately predict the velocity of drifters. They used the drifter data obtained from the Gulf of Mexico. Their model was much more accurate than the model using the Maxey–Riley equation [26] in terms of root mean square error (RMSE).

3. Data

Our previous study [8] used data obtained from Seosan, located on the west coast of South Korea. For this work, we added new data obtained from Jeju Island to the previous data. Hourly data from 2015 to 2016 at these locations were used for our study. We used this to predict the hourly locations of drifters in this study. Figure 1 shows the observed trajectories from the two locations in 2015, and Table 1 shows the features related to the start and end points of trajectories of the sample data. The height of the wind velocity above the ocean was 10 m, and its spatial resolution was 900 m.

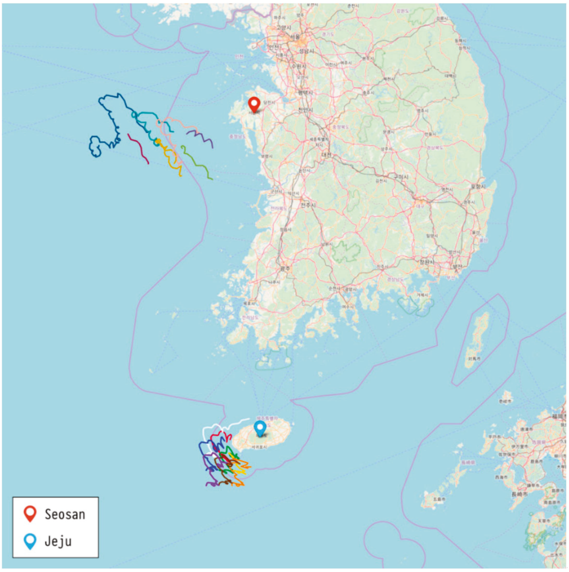


Figure 1. Observed trajectory data from Seosan and Jeju in 2015.

Table 1. Attributes of the data (examples).

Location	Case No.	#Data	Start						End					
			Y	M	D	H	Lon.	Lat.	Y	M	D	H	Lon.	Lat.
Seosan	1	239	2015	11	6	8	125.0795	36.5788	2015	11	16	6	124.4480	36.4438
	2	26	2015	11	6	9	125.0858	36.2526	2015	11	7	13	124.8229	36.5638
	5	109	2015	11	6	15	125.4192	36.2578	2015	11	11	3	125.5130	36.1293
	6	112	2015	11	6	16	125.4081	36.5825	2015	11	11	7	125.1982	36.4487
	7	113	2015	11	6	17	125.7388	36.5810	2015	11	11	9	125.4705	36.2202
	8	39	2015	11	6	18	125.9147	36.4168	2015	11	8	8	125.5818	36.5624
	9	45	2015	11	6	13	125.9123	36.0874	2015	11	8	9	125.5238	36.4135
Jeju	1	82	2016	4	18	23	126.0006	33.1695	2016	4	22	8	126.1397	33.3354
	2	106	2016	4	18	23	125.8375	33.1774	2016	4	23	8	126.3886	33.5580
	3	106	2016	4	18	22	125.8338	33.0041	2016	4	23	7	126.0698	33.3213
	4	87	2016	4	19	0	126.0045	33.0026	2016	4	22	14	126.2295	33.1773
	5	60	2016	4	19	0	126.1655	33.9999	2016	4	21	11	126.1255	33.1871
	6	93	2016	4	18	20	126.1671	32.8383	2016	4	22	16	126.4106	33.0411
	7	94	2016	4	18	21	125.9990	32.8439	2016	4	22	18	12.2667	33.0728
	8	93	2016	4	18	21	125.8346	32.8395	2016	4	22	17	126.0957	33.0810

4. Discussion

This study added machine learning (ML) techniques to the methods of our prior work [8]. We used regression functions for numerical predictions, and also examined various artificial neural networks.

4.1. Numerical Model and Evolutionary Methods

We used MOHID [6] as a numerical model using the Navier-Stokes equations [27], and we also examined various evolutionary methods from our previous study [8]. The evolutionary methods include differential evolution (DE) [28], particle swarm optimization (PSO) [29], and the covariance matrix adaptation evolutionary strategy (CMA-ES) [30].

4.2. Machine Learning

We used supervised learning to find the mapping between input data (wind velocity and flow velocity) and output data (drifter location). The performance of the following two regression functions were examined.

Support vector regression (SVR): While conventional classifiers minimize error rates during the training process, support vector machines (SVMs) construct a set of hyperplanes so that the distance from it to the nearest training data point is maximized. They were considered as alternatives to artificial neural networks in the 1990s since nonlinear classification became possible through the kernel trick [31]. Support vector regression uses SVM for regression with continuous values as the output [32].

Gaussian process (GP): GP [33] is an ML model that predicts data as the average and variance of probability distributions. It predicts functions that can represent given data in the defined function distribution and estimates functions for experimental data based on the arbitrary training data.

4.3. Artificial Neural Networks

Artificial neural networks represent a learning algorithm inspired by the neural networks of biology. With the recent advancement of deep learning, the use of artificial neural networks has yielded excellent performance in classification and can be used for regression when a mean-square error (MSE) is the loss function. Below are the neural network methods we used in this study.

Multi-layer perceptron (MLP) is a basic neural network structure that adds hidden layers into the perception structure. We built a hierarchical model with four inputs, two outputs, and one hidden layer.

Radial basis function network (RBFN) is a type of neural network that represents the proximity to the correct answer using Gaussian probability and Euclidian distance [34]. One hidden layer based on Gaussian probability distribution is used, and the training process is extremely fast.

Deep neural network (DNN) increases training parameters by adding hidden layers in MLP. Figure 2 shows the settings of the input and output values in the basic DNN structure. It is often essential to use a rectified linear unit (ReLU) [35] and Dropout [36] to prevent the vanishing gradient problem [37].

Recurrent neural network (RNN): the connection between units is characterized by a circular structure [38] and can be used when continuous data is given. RNNs can be used to predict trajectories since the movement of a drifter is sequential data. The input data for MLP or DNN has a fixed size of 4, as depicted in Figure 2. However, RNNs should incorporate all the drifter moves in sequence; thus, the data length can be different. The maximum length that the model can receive as an input sequence is set, and the remaining data space is padded with zero. For example, if the maximum length is set to 200 and given data set has 120 h of movement information, the remaining data is filled with 80 zeros. Figure 3 shows the input and output data structure of the RNN model.

Long short-term memory (LSTM): the RNN model creates the next unit based on product operation and suffer from the vanishing gradient problem when dealing with long data sequences. LSTM [39], which has the function of forgetting past information and of remembering current information by adding a cell-state to the hidden state of RNN, has emerged as one of the most widely used RNN methods and can effectively solve not only the vanishing gradient problem but the long-term dependency. This model is expected to remember the movement of a drifter according to the specific short sequence of wind and flow.

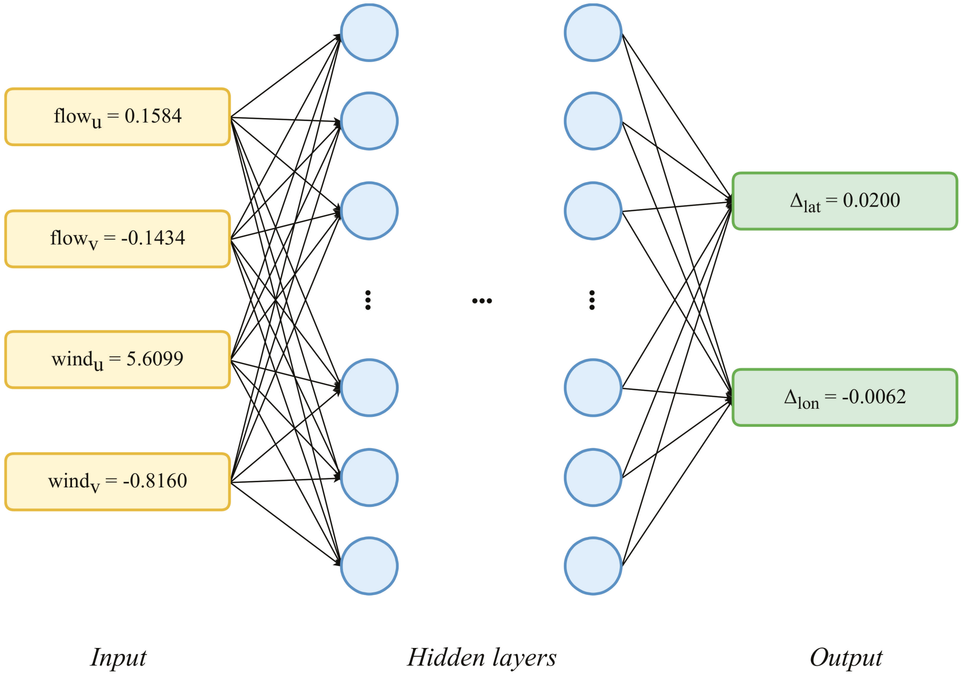


Figure 2. Data input/output in our deep neural network (DNN) model.

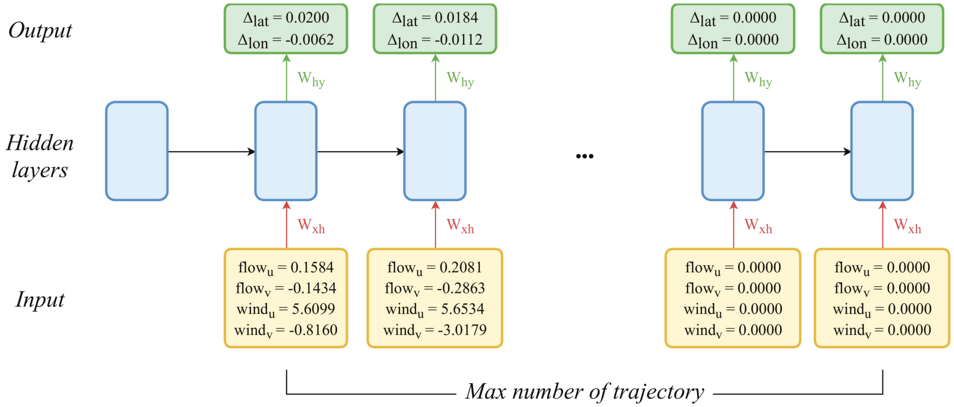

Output
Hidden layers
Input
$\Delta_{lat} = 0.0200$ $\Delta_{lon} = -0.0062$
$\Delta_{lat} = 0.0184$ $\Delta_{lon} = -0.0112$
$\Delta_{lat} = 0.0000$ $\Delta_{lon} = 0.0000$
$\Delta_{lat} = 0.0000$ $\Delta_{lon} = 0.0000$
W_{hy}
W_{xh}
$\text{flow}_u = 0.1584$ $\text{flow}_v = -0.1434$ $\text{wind}_u = 5.6099$ $\text{wind}_v = -0.8160$
$\text{flow}_u = 0.2081$ $\text{flow}_v = -0.2863$ $\text{wind}_u = 5.6534$ $\text{wind}_v = -3.0179$
$\text{flow}_u = 0.0000$ $\text{flow}_v = 0.0000$ $\text{wind}_u = 0.0000$ $\text{wind}_v = 0.0000$
$\text{flow}_u = 0.0000$ $\text{flow}_v = 0.0000$ $\text{wind}_u = 0.0000$ $\text{wind}_v = 0.0000$
Max number of trajectory

Figure 3. Data input/output in our recurrent neural network (RNN) model.

5. Experiments

5.1. Setting and Environments

We implemented the evolutionary computation methods using DEAP software (https://deap.readthedocs.io/en/master/) except for PSO. For PSO, we used PySwarms (https://pyswarms.readthedocs.io/en/latest/), which performed better than DEAP. We used WEKA 3 [40] for MLP, GP, SVR, and RBFN, and PyTorch [41] for DNN, RNN, and LSTM. Table 2 summarizes software libraries we used.

The previous methods of evolutionary computation evaluated test data by creating a single model per method. When a single method yields several models, the performance may vary depending on the random seed. We confirmed that there are considerable performance differences between models using the same method for deep learning methods. We used the bagging [42] method to analyze each method by creating 10 models for each method and then measuring the average value, variance, and standard deviation of the resulting measured values. There might be sharp loss value changes in the process of solving local minimum problems for deep learning methods. We calculated mean absolute error (MAE) according to epoch and ended training when the MAE value was low in our experiments.

Table 2. Methods and library resources.

Method	Library
Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES)	DEAP
Differential Evolution (DE)	DEAP
Particle Swarm Optimization (PSO)	PySwarms
Multi-Layer Perceptron MLP	WEKA 3
Gaussian Process (GP)	WEKA 3
Support Vector Regression (SVR)	WEKA 3
Radial Basis Function Network (RBFN)	WEKA 3
Deep Neural Network (DNN)	PyTorch
Recurrent Neural Network (RNN)	PyTorch
Long Short-Term Memory (LSTM)	PyTorch

5.2. Evaluation Measures

We used mean absolute error (MAE) and normalized cumulative Lagrangian separation (NCLS). In prior work [8], we also used the Euclidean distance as an evaluation measure. Since it is calculated by a mechanism similar to MAE, the average of the error distance was calculated only using MAE. NCLS, also called the skill score, was calculated by subtracting the error from 1. Lower values, therefore, represent better results for MAE, whereas higher values close to 1 represent better results for NCLS. The calculation for MAE can be expressed as the following Equation.

$$\frac{1}{m}\sum_{i=1}^{m}(|\,pred\_Lat_i\ -\ observed\_Lat_i| + |pred\_Lon_i\ -\ observed\_Lon_i|) \tag{1}$$

where *pred_Lon* and *pred_Lat* represent the longitude and latitude of the predicted data. *Observed_Lon* and *observed_Lat* denote the longitude and the latitude of the observed data. Therefore, the difference between these values can be considered as an error. The denominator m is the number of test datasets. Therefore, the MAE is the average of the errors.

NCLS is a measurement method that is quite frequently used in trajectory modeling and it is proposed to solve weaknesses in the Lagrangian separation distance in relation to the continental shelf and its adjacent deep ocean. The error of each location is calculated in MAE, whereas NCLS calculates errors by cumulative calculation. Figure 4 shows the calculation process of NCLS. In this process, if s becomes too large, the skill score may continue to remain zero. If the tolerance threshold n is set high, this can be solved to some extent. In this study, we set n to 1 since errors sufficient to make s relatively large did not frequently occur.

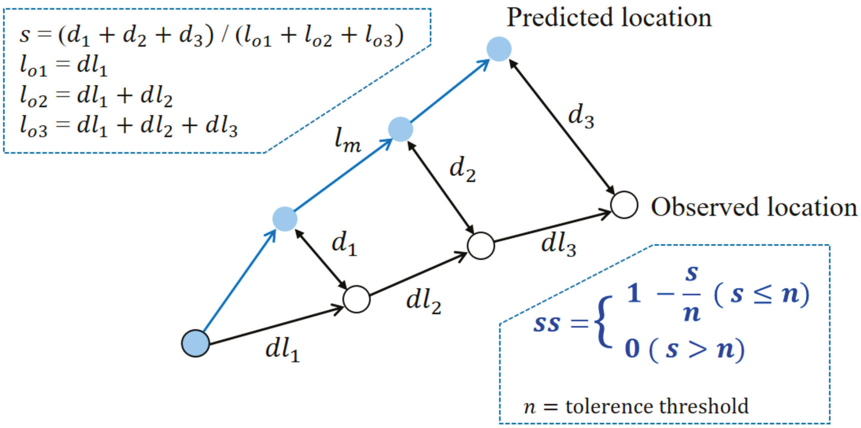

$s = (d_1 + d_2 + d_3) / (l_{o1} + l_{o2} + l_{o3})$
$l_{o1} = dl_1$
$l_{o2} = dl_1 + dl_2$
$l_{o3} = dl_1 + dl_2 + dl_3$
Predicted location
l_m
d_3
d_2
d_1
Observed location
dl_1
dl_2
dl_3
$ss = \begin{cases} 1 - \frac{s}{n} \ (s \le n) \\ 0 \ (s > n) \end{cases}$
n = tolerence threshold

Figure 4. Formula to calculate skill score (*ss*) of NCLS.

5.3. Results

Previous measurements are available for evolutionary computational methods. In Section 5.3.1, we verified the performance of the Python-based system by comparing the results with only the Seosan data. The degree of training is also an essential factor. Prior to experimenting with all the data, in Section 5.3.2, we measured epoch numbers deemed good enough to end training by measuring MAE for each epoch in each deep learning method. Lastly, in Section 5.3.3, we trained them using all the data and described the experimental results that predicted the trajectories of the newly added Jeju data.

5.3.1. Evolutionary Search on Seosan Data

Table 3 compares the results of the previous study (C language) [8] and this study (Python). CMA-ES showed particularly good performance compared to the previous study. Overall, we could improve the performance of evolutionary search by using new software libraries.

Table 4 shows the CPU time to build the prediction models for Seosan data. Since inference time is usually much shorter than training time, actual performance can be more important than training speed.

Table 3. Parameters of evolutionary computation methods.

Method	Evaluation	Case			
		1	5	6	7
DE (Previous)	MAE	0.0920	0.0828	0.0392	0.0653
	NCLS	0.9026	0.7965	0.9220	0.8826
DE	MAE	0.0587	0.0973	0.0380	0.0568
	NCLS	0.9451	0.7686	0.9225	0.8974
PSO (Previous)	MAE	0.0907	0.0820	0.0385	0.0622
	NCLS	0.9044	0.7984	0.9232	0.8878
PSO	MAE	0.0603	0.0973	0.0380	0.0567
	NCLS	0.9441	0.7686	0.9226	0.8976
CMA-ES (Previous)	MAE	0.1303	0.1393	0.0761	0.1304
	NCLS	0.8631	0.6513	0.8437	0.7297
CMA-ES	MAE	0.0582	0.0973	0.0380	0.0567
	NCLS	0.9457	0.7686	0.9226	0.8976
MOHID model [10]	MAE	0.1352	0.1238	0.0656	0.0434
	NCLS	0.8633	0.7134	0.8480	0.9229

The lower the mean absolute error (MAE) values, the better. The higher the NCLS values, the better.

Table 4. Computing time of evolutionary computation methods.

Data		Case 1	Case 5	Case 6	Case 7
Computing time (CPU second)	DE	2120.8	2705.8	2663.2	3066.2
	PSO	302.5	345.7	344.9	371.5
	CMA-ES	117.5	152.8	152.9	173.9

5.3.2. Deep Learning

The neural network methods use the loss value to calculate how well a model is trained. The loss value decreases as the model accurately predicts the training data. It is better to use cross-entropy and *softmax* in the final layer of neural network-based classifiers [43]. However, we used MSE as the loss function since we predicted continuous values, not discrete ones.

The loss function measures the difference between the correct answer of the training data and the value predicted by the model, which may not relate to MSE and NCLS. In order to identify whether or not loss and MAE are related to each other, we examined several neural network models. Neural networks learn the values of the weights to reduce loss, and a reduction of MSE can prove worthwhile. We investigated the loss according to epoch, MAE, and MAE of the test data in the three deep learning methods, CNN, RNN, and LSTM. From Jeju data, Case 1 was used as the test data, and the rest were used as training data. Figure 5 shows the results of DNN.

DNN calculates the error between individual data independently. As the training progresses, the loss value generally decreases. However, the MAE of the training data and test data decreases sharply only at the beginning, and the performance does not significantly improve thereafter. We looked for an additional way to solve this problem, since the MAE deviation between each epoch is large even when the training is complete. We attempted to solve the problem by bagging among the ensemble techniques. The final epoch of DNN is set to 1500. The MAE of the test data was low in the 100–200 epoch section when training was incomplete, and the MAE deviation between epochs was large even post-training in the case of DNN. After 1500 epochs, the loss value did not decrease further, so we set the final epoch of DNN to 1500.

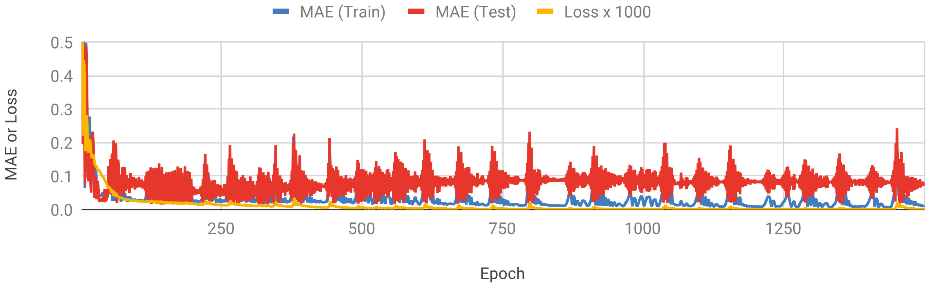


Figure 5. MAE and Loss of DNN.

Figure 6 shows the MAE and loss values of RNN. The data are continuous time-series data of the movement of a drifter over time. Although the loss value is reduced above 100 epochs, the MAE of the training and test data did not decrease. We set the final epoch of RNN to 1000. LSTM was similar to RNN, but there were ups and downs on the MAE graph as learning progressed. The final epoch of the LSTM was set to 500. Figure 7 shows the MAE and loss values of LSTM.

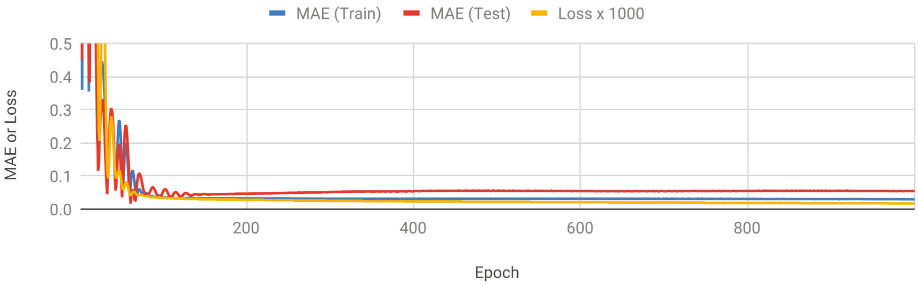


Figure 6. MAE and loss values of RNN.

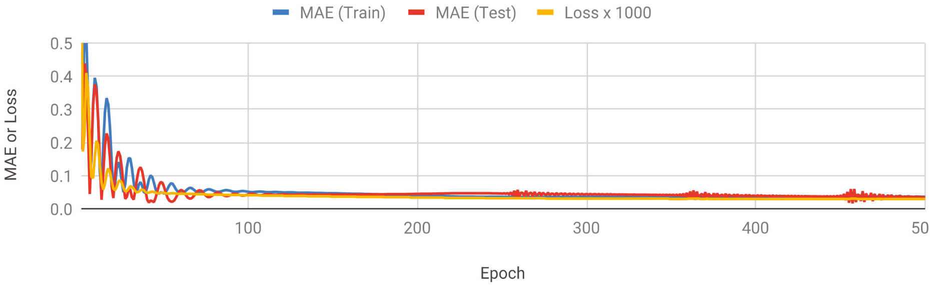


Figure 7. MAE and loss values of LSTM.

5.3.3. Results for Each Case

We describe the performance measurement results of methods beyond the evolutionary computation mentioned in Section 3. Table 5 shows the main parameter values for each method. Our prior work [8] measured the error per iteration. The deep leaning method using PyTorch (e.g., DNN, RNN, and LSTM) is conceptually similar; MAE and loss were measured per epoch. The result based on the WEKA library could not measure the error according to the iteration since it could not set the iteration number internally.

Table 6 shows the evaluation of measured values by building models from ML and evolutionary computation methods. Only the Seosan data was used for training. GP, MLP, RBFN, and SVR based on WEKA showed outstanding performance. However, the deep learning methods did not perform very well as the variation in data volume by case was large for the Seosan data. The difference between the numbers of Cases 1 and 2 was nine times, as shown in Table 1. The data length also has a significant impact on training because the previous event influences the next event in RNN. Table 7 shows the CPU time spent building models for each method. Evolutionary computation methods took more time than GP, MLP, and RFBN using WEKA software.

Table 8 shows the experiment results with the Jeju data. Unlike the result of the Seosan data, the deep learning methods showed excellent performance. LSTM performed better than the basic RNN models, and sometimes DNN performed better. In Cases 1, 2, and 3, MLP and RBFN using WEKA software indicated good performance, whereas the evolutionary computation methods are neither good nor bad performance. The results of the RNN methods could be improved since the variation of the number of data by case is small for the Jeju data. Table 9 shows the CPU time for calculation. The computation time for the Jeju data was not much different from that of the Seosan data.

Table 5. Parameter values for machine learning (ML) methods.

Method	Parameter	Value
GP (WEKA 3)	Filter type Level of Gaussian noise Decimal places	Standardized training data 1.0 2
MLP (WEKA 3)	Activation function Loss function Decimal places Number of hidden units Ridge Tolerance	Approximate sigmoid Square error 2 1 0.01 1.0×10^{-6}
RBFN (WEKA 3)	Decimal places Number of basis functions Ridge Scale optimization Tolerance	2 2 2 Use scale per unit 1.0×10^{-6}
SVR (WEKA 3)	Complexity constant Filter type Kernel Decimal places	3.0 Standardized training data Polynomial kernel 2
DNN (PyTorch)	Loss function Activation function Number of epochs Number of hidden layers Number of units per a layer	Mean square error ReLU 1500 8 256
RNN (PyTorch)	Loss function Number of hidden units Number of epochs	Mean square error 256 1000
LSTM (PyTorch)	Loss function Number of hidden units Number of epochs	Mean square error 256 500

Table 6. Results for Seosan data.

Method	Evaluation	Case			
		1	5	6	7
DE	MAE	0.0743	0.0286	0.0906	0.0417
	NCLS	0.9308	0.7377	0.7769	0.9172
PSO	MAE	0.1238	0.0318	0.1050	0.0758
	NCLS	0.8724	0.7183	0.7557	0.8375
CMA-ES	MAE	0.0743	0.0286	0.0909	0.0418
	NCLS	0.9308	0.7378	0.7763	0.9170
GP	MAE	0.0945	0.0290	0.0974	0.0360
	NCLS	0.9113	0.7431	0.7583	0.9282
MLP	MAE	0.0957	0.0285	0.1012	0.0320
	NCLS	0.9030	0.7541	0.7474	0.9353
RBFN	MAE	0.1064	0.0278	0.0986	0.0297
	NCLS	0.8992	0.7626	0.7508	0.9399
SVR	MAE	0.0700	0.0301	0.0939	0.0394
	NCLS	0.9355	0.7402	0.7711	0.9204
DNN	MAE	0.1557	0.0192	0.1305	0.0571
	NCLS	0.8493	0.8383	0.6879	0.8848
RNN	MAE	0.1369	0.0270	0.1335	0.0494
	NCLS	0.8555	0.7699	0.6809	0.8910
LSTM	MAE	0.0744	0.0295	0.1153	0.0301
	NCLS	0.9201	0.7413	0.7075	0.9408
MOHID	MAE	0.1352	0.1238	0.0656	0.0434
	NCLS	0.8633	0.7134	0.8480	0.9229

The lower the MAE values, the better. The higher the NCLS values, the better.

Table 7. Computing time for Seosan data.

Data		Case 1	Case 5	Case 6	Case 7
Computing time (CPU second)	DE	2120.78	2705.83	2663.18	3066.16
	PSO	302.53	345.69	344.87	371.54
	CMA-ES	117.53	152.80	152.93	173.92
	GP	9.12	13.62	14.78	13.82
	MLP	4.45	4.33	4.50	4.60
	RBFN	3.69	3.72	3.77	3.86
	SVR	9.57	15.00	13.89	14.37
	DNN	153.36	182.26	174.52	178.29
	RNN	45.93	47.58	44.56	48.27
	LSTM	34.47	45.59	44.32	46.46

Table 8. Results for Jeju data.

Method	Evaluation	Case							
		1	2	3	4	5	6	7	8
DE	MAE	0.0497	0.0703	0.0500	0.0576	0.0337	0.0460	0.0495	0.0638
	NCLS	0.8685	0.8698	0.8930	0.8675	0.8770	0.8763	0.8805	0.8594
PSO	MAE	0.0491	0.0699	0.0501	0.0578	0.0331	0.0461	0.0480	0.0646
	NCLS	0.8702	0.8700	0.8927	0.8671	0.8785	0.8760	0.8855	0.8576
CMA-ES	MAE	0.0491	0.0702	0.0502	0.0575	0.0338	0.0461	0.0494	0.0640
	NCLS	0.8700	0.8699	0.8927	0.8677	0.8767	0.8761	0.8807	0.8589
GP	MAE	0.0447	0.0468	0.0450	0.0701	0.0273	0.0583	0.0676	0.0836
	NCLS	0.8912	0.9096	0.9115	0.8420	0.8981	0.8466	0.8439	0.8202
MLP	MAE	0.0345	0.0571	0.0398	0.0586	0.0271	0.0599	0.0669	0.0816
	NCLS	0.9145	0.8883	0.9191	0.8663	0.8988	0.8488	0.8482	0.8224
RBFN	MAE	0.0468	0.0418	0.0382	0.0632	0.0268	0.0493	0.0564	0.0723
	NCLS	0.8871	0.9192	0.9249	0.8583	0.8996	0.8727	0.8737	0.8438
SVR	MAE	0.0378	0.0542	0.0442	0.0581	0.0335	0.0507	0.0495	0.0721
	NCLS	0.9073	0.8970	0.9099	0.8683	0.8769	0.8607	0.8801	0.8432
DNN	MAE	0.0453	0.0931	0.0384	0.0487	0.0364	0.0459	0.0352	0.0436
	NCLS	0.8850	0.8193	0.9218	0.8823	0.8629	0.8849	0.9164	0.9007
RNN	MAE	0.0483	0.0635	0.0554	0.0595	0.0506	0.0661	0.0414	0.0647
	NCLS	0.8787	0.8710	0.8840	0.8510	0.8226	0.8346	0.8935	0.8504
LSTM	MAE	0.0478	0.0608	0.0526	0.0445	0.0338	0.0362	0.0411	0.0451
	NCLS	0.8841	0.8812	0.8907	0.8953	0.8793	0.9071	0.9076	0.9012
MOHID	MAE	0.0622	0.0630	0.0883	0.1043	0.0236	0.0724	0.0864	0.1116
	NCLS	0.8361	0.8792	0.8271	0.7559	0.9140	0.8324	0.8007	0.7585

Table 9. Computing time for Jeju data.

Data		Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Computing time (CPU second)	DE	3004.0	2879.6	2900.4	2956.5	3098.1	2966.9	2978.2	2958.7
	PSO	366.6	361.0	360.8	365.5	374.6	363.9	362.9	365.6
	CMA-ES	165.9	158.9	160.8	165.2	171.5	162.0	161.2	162.9
	GP	7.04	6.47	6.49	6.93	7.64	6.75	6.79	6.86
	MLP	1.61	1.63	1.59	1.59	1.63	1.62	1.53	1.58
	RFBN	1.56	1.58	1.51	1.50	1.52	1.61	1.53	1.53
	SVR	3.36	3.45	3.22	3.28	3.66	3.35	3.56	3.25
	DNN	172.12	174.94	170.06	169.84	174.01	168.76	169.10	164.17
	RNN	45.75	50.57	46.14	49.22	52.09	49.34	49.45	47.43
	LSTM	55.30	49.49	53.18	46.52	51.44	48.17	47.93	45.26

5.3.4. Weighted Average Results

We compared evolutionary algorithms and ML using the Seosan and Jeju data. However, it was not easy to find out which method was better overall. We used the weight-averaged results and trajectory plots of predicted and actual points. The weighted average provides an advantage when the number of data for each case is different. Experiments with more data are considered more important; thus weights are based on the number of data. The weighted average is calculated as follows:

$$\sum\nolimits_{i=1}^{n} d_i r_i \Big/ \sum\nolimits_{i=1}^{n} d_i, \tag{2}$$

where d_i refers to the number of data of the ith case and r_i refers to the evaluation result of the ith case. One of the indicators covered in this section is the standard deviation. As described in Section 4.1, this experiment uses the method of building ten models, evaluating each of them, and then obtaining the average. The performance of each model may vary for each run. For practical use, we need to determine whether or not the performance deviation of each model is large.

Table 10 shows the overall performance of each method. In this context, CMA-ES showed the highest performance on the Seosan data and LSTM on the Jeju data. The performance of DNN and RNN on the Jeju data was good. The amount of data has to be equalized in each case of using neural networks. Rankings were calculated separately for each method in terms of MAE and NCLS. RBFN was the best for MAE, and LSTM was the best for NCLS. Since NCLS is more popular measure for predictions of drifter trajectory than MAE, LSTM was the best method for this study. As expected, LSTM was superior to RNN. There is past information to be forgotten and current information to be remembered according to the direction of the drifters. The performance of DNN and RNN was good for Jeju, so if we get more data in the future, it will be possible improve their performance. Evolutionary methods showed good performance on both Seosan and Jeju data. Especially for Seosan, where the number of data in each case was large, these methods showed better performance than the other methods.

Table 10. Results of weighted average and standard deviation values.

Method	Evaluation	Seosan		Jeju		Integration		
		Weighted Average	Standard Deviation	Weighted Average	Standard Deviation	Weighted Average	Standard Deviation	Rank
DE	MAE	0.0660	0.0001	0.0528	0.0000	0.0567	0.0001	3
	NCLS	0.8634	0.0001	0.8712	0.0001	0.8717	0.0002	5
PSO	MAE	0.0979	0.0293	0.0642	0.0068	0.0569	0.0010	4
	NCLS	0.8069	0.0386	0.8418	0.0234	0.8717	0.0024	6
CMA-ES	MAE	0.0660	0.0002	0.0529	0.0001	0.0567	0.0000	2
	NCLS	0.8635	0.0004	0.8709	0.0004	0.8718	0.0000	4
GP	MAE	0.0743	0.0000	0.0554	0.0000	0.0589	0.0000	8
	NCLS	0.8541	0.0000	0.8682	0.0000	0.8675	0.0000	8
MLP	MAE	0.0747	0.0033	0.0530	0.0013	0.0584	0.0007	7
	NCLS	0.8515	0.0042	0.8741	0.0036	0.8686	0.0010	7
RBFN	MAE	0.0793	0.0062	0.0435	0.0012	0.0556	0.0018	1
	NCLS	0.8493	0.0062	0.8989	0.0027	0.8753	0.0037	2
SVR	MAE	0.0662	0.0000	0.0527	0.0000	0.0576	0.0000	5
	NCLS	0.8613	0.0000	0.8724	0.0000	0.8731	0.0000	3
DNN	MAE	0.1100	0.0175	0.0465	0.0057	0.0733	0.0121	9
	NCLS	0.7986	0.0250	0.8915	0.0141	0.8487	0.0192	9
RNN	MAE	0.0993	0.0223	0.0416	0.0052	0.0958	0.0326	10
	NCLS	0.8083	0.0259	0.9008	0.0143	0.8065	0.0516	10
LSTM	MAE	0.0702	0.0007	0.0411	0.0004	0.0579	0.0035	6
	NCLS	0.8506	0.0017	0.9040	0.0010	0.8762	0.0059	1
MOHID	MAE	0.0932	–	0.0789	–	0.0861	–	–
	NCLS	0.8201	–	0.8228	–	0.8215	–	–

Figure 8 shows the trajectory of a drifter predicted by CMA-ES, which showed the best performance for Seosan. Figure 9 exhibits the trajectory of a drifter predicted by LSTM, which had the best performance for Jeju. Finally, Figure 10 presents the trajectory of a drifter predicted by RBFN with the best performance for MAE. Both ML and evolutionary search optimize the parameters. There is a slight difference in accuracy, but all of them predict similar paths.

Except for the RNN series (RNN and LSTM), only the data at that point were used to predict the trajectory of a drifter at a point in hourly time. However, there is a difference in that RNN uses hidden layer neurons, which were used for prediction in previous instances, and LSTM showed the best performance as a measure of NCLS. It is believed that this is because the RNN series take into account the inertia of the drifter.

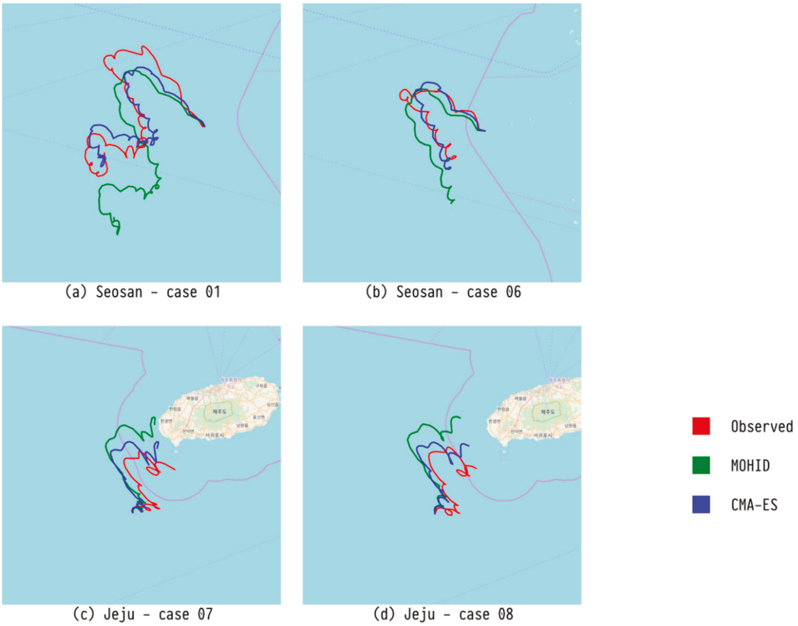
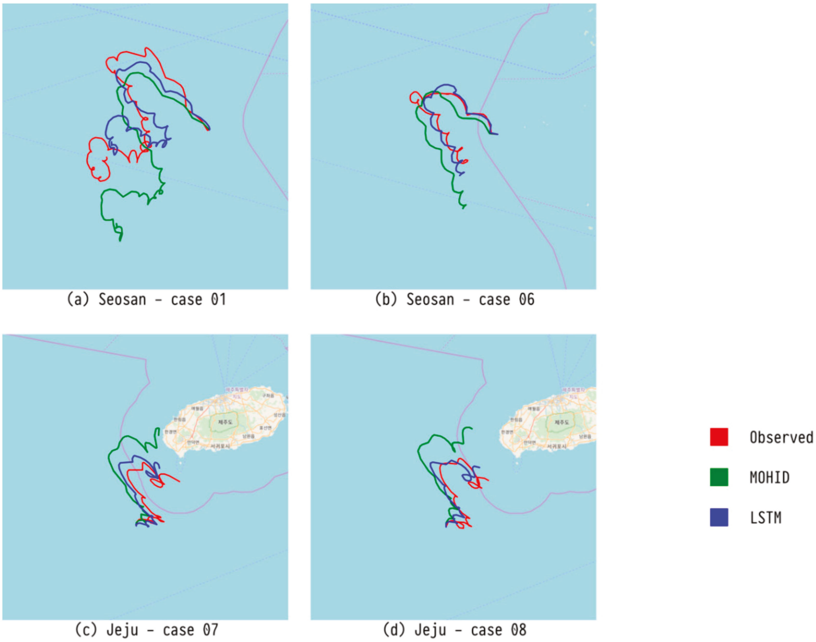


Figure 8. Comparison of trajectory predicted by our CMA-ES model, trajectory predicted by an existing numerical model (MOHID), and observed trajectory for four major drifters.



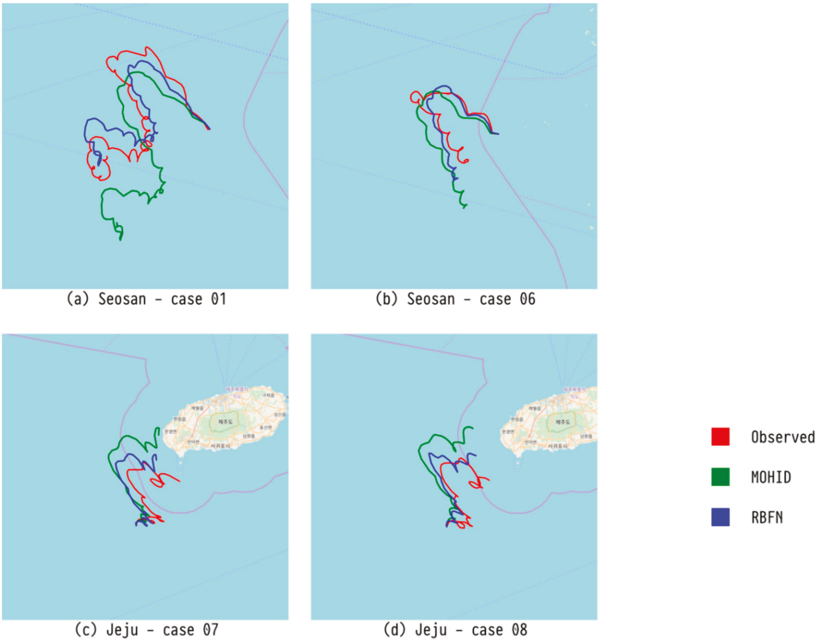
(a) Seosan - case 01

(b) Seosan - case 06

(c) Jeju - case 07

(d) Jeju - case 08

Figure 9. Comparison of trajectory predicted by our LSTM model, trajectory predicted by an existing numerical model (MOHID), and observed trajectory for four major drifters.



(a) Seosan - case 01

(b) Seosan - case 06

(c) Jeju - case 07

(d) Jeju - case 08

Figure 10. Comparison of trajectory predicted by our RBFN model, trajectory predicted by an existing numerical model (MOHID), and observed trajectory for four major drifters.

6. Conclusions

We extended and improved our previous study [8] which predicted the trajectories of drifters using evolutionary computation, and we also predicted the trajectories of drifters using various machine learning techniques [44–49]. To the best of the authors' knowledge, this was the first trial in which machine learning has been applied to the prediction of drifter trajectories, and it significantly improved upon the representative numerical model, MOHID.

In terms of MAE, RBFN using the WEKA library showed the best performance, an improvement of 35.20% over the numerical model MOHID. LSTM using PyTorch showed the best performance regarding NCLS, an improvement of 6.24% over MOHID. These neural network-based methods did not take a long time to construct a model. In the future, we plan to experiment with other representative variants of RNNs such as gated recurrent units [50], and we will design more models that increase the performance of DNNs or basic RNNs by adding more training data.

Author Contributions: Conceptualization, D.-Y.K. and Y.-H.K.; methodology, Y.-W.N. and H.-Y.C.; software, Y.-W.N. and H.-Y.C.; validation, Y.-W.N., H.-Y.C. and D.-Y.K.; formal analysis, Y.-W.N. and Y.-H.K.; investigation, Y.-H.K.; resources, D.-Y.K.; data curation, D.-Y.K.; writing—original draft preparation, Y.-W.N., H.-Y.C., S.-H.M. and Y.-H.K.; writing—review and editing, S.-H.M. and Y.-H.K.; visualization, H.-Y.C.; supervision, Y.-H.K.; project administration, Y.-H.K.; funding acquisition, D.-Y.K. and Y.-H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was a part of the project titled 'Marine Oil Spill Risk Assessment and Development of Response Support System through Big Data Analysis', funded by the Ministry of Oceans and Fisheries, Korea.

Conflicts of Interest: The authors declare that there is no conflict of interest regarding the publication of this article.

References

1. Nasello, C.; Armenio, V. A New Small Drifter for Shallow Water Basins: Application to the Study of Surface Currents in the Muggia Bay (Italy). *J. Sens.* **2016**, *2016*, 1–5. [CrossRef]
2. Sayol, J.M.; Orfila, A.; Simarro, G.; Conti, D.; Renault, L.; Molcard, A. A Lagrangian model for tracking surface spills and SaR operations in the ocean. *Environ. Model. Softw.* **2014**, *52*, 74–82. [CrossRef]
3. Sorgente, R.; Tedesco, C.; Pessini, F.; De Dominicis, M.; Gerin, R.; Olita, A.; Fazioli, L.; Di Maio, A.; Ribotti, A. Forecast of drifter trajectories using a Rapid Environmental Assessment based on CTD observations. *Deep Sea Res. II Top. Stud. Oceanogr.* **2016**, *133*, 39–53. [CrossRef]
4. Zhang, W.-N.; Huang, H.-M.; Wang, Y.-G.; Chen, D.-K.; Zhang, L. Mechanistic drifting forecast model for a small semi-submersible drifter under tide–wind–wave conditions. *China Ocean Eng.* **2018**, *32*, 99–109. [CrossRef]
5. De Dominicis, M.; Pinardi, N.; Zodiatis, G.; Archetti, R. MEDSLIK-II, a Lagrangian marine surface oil spill model for short-term forecasting—Part 2: Numerical simulations and validation. *Geosci. Model Dev.* **2013**, *6*, 1999–2043. [CrossRef]
6. Miranda, R.; Braunschweig, F.; Leitao, P.; Neves, R.; Martins, F.; Santos, A. MOHID 2000—A coastal integrated object oriented model. *WIT Trans. Ecol. Environ.* **2000**, *40*, 1–9.
7. Beegle-Krause, J. General NOAA oil modeling environment (GNOME): A new spill trajectory model. *Int. Oil Spill Conf.* **2001**, *2001*, 865–871. [CrossRef]
8. Nam, Y.-W.; Kim, Y.-H. Prediction of drifter trajectory using evolutionary computation. *Discrete Dyn. Nat. Soc.* **2018**, *2018*, 1–15. [CrossRef]
9. Lee, C.-J.; Kim, G.-D.; Kim, Y.-H. Performance comparison of machine learning based on neural networks and statistical methods for prediction of drifter movement. *J. Korea Converg. Soc.* **2017**, *8*, 45–52. (In Korean)
10. Lee, C.-J.; Kim, Y.-H. Ensemble design of machine learning techniques: Experimental verification by prediction of drifter trajectory. *Asia-Pac. J. Multimed. Serv. Converg. Art Humanit. Sociol.* **2018**, *8*, 57–67. (In Korean)
11. Özgökmen, T.M.; Piterbarg, L.I.; Mariano, A.J.; Ryan, E.H. Predictability of drifter trajectories in the tropical Pacific Ocean. *J. Phys. Oceanogr.* **2001**, *31*, 2691–2720. [CrossRef]
12. Belore, R.; Buist, I. Sensitivity of oil fate model predictions to oil property inputs. In Proceedings of the Arctic and Marine Oilspill Program Technical Seminar, Vancouver, BC, Canada, 8–10 June 1994.

13. Lehr, W.; Jones, R.; Evans, M.; Simecek-Beatty, D.; Overstreet, R. Revisions of the adios oil spill model. *Environ. Model. Softw.* **2002**, *17*, 189–197. [CrossRef]
14. Berry, A.; Dabrowski, T.; Lyons, K. The oil spill model OILTRANS and its application to the Celtic Sea. *Mar. Pollut. Bull.* **2012**, *64*, 2489–2501. [CrossRef] [PubMed]
15. Applied Science Associates. *OILMAP for Windows (Technical Manual)*; ASA Inc.: Narranganset, RI, USA, 1997.
16. Reed, M.; Singsaas, I.; Daling, P.S.; Faksnes, L.-G.; Brakstad, O.G.; Hetland, B.A.; Hofatad, J.N. Modeling the water-accommodated fraction in OSCAR2000. *Int. Oil Spill Conf.* **2001**, *2001*, 1083–1091. [CrossRef]
17. Al-Rabeh, A.; Lardner, R.; Gunay, N. Gulfspill Version 2.0: A software package for oil spills in the Arabian Gulf. *Environ. Model. Softw.* **2000**, *15*, 425–442. [CrossRef]
18. Pierre, D. Operational forecasting of oil spill drift at Meétéo-France. *Spill Sci. Technol. Bull.* **1996**, *3*, 53–64. [CrossRef]
19. Annika, P.; George, T.; George, P.; Konstantinos, N.; Costas, D.; Koutitas, C. The Poseidon operational tool for the prediction of floating pollutant transport. *Mar. Pollut. Bull.* **2001**, *43*, 270–278. [CrossRef]
20. Zodiatis, G.; Lardner, R.; Solovyov, D.; Panayidou, X.; De Dominicis, M. Predictions for oil slicks detected from satellite images using MyOcean forecasting data. *Ocean Sci.* **2012**, *8*, 1105–1115. [CrossRef]
21. Hackett, B.; Breivik, Ø.; Wettre, C. Forecasting the drift of objects sand substances in the ocean. In *Ocean Weather Forecasting: An Integrated View of Oceanography*; Springer: Dordrecht, The Netherlands, 2006.
22. Breivik, Ø.; Allen, A.A. An operational search and rescue model for the Norwegian Sea and the North Sea. *J. Mar. Syst.* **2008**, *69*, 99–113. [CrossRef]
23. Broström, G.; Carrasco, A.; Hole, L.; Dick, S.; Janssen, F.; Mattsson, J.; Berger, S. Usefulness of high resolution coastal models for operational oil spill forecast: The Full City accident. *Ocean Sci. Discuss.* **2011**, *8*, 1467–1504. [CrossRef]
24. Ambjörn, C.; Liungman, O.; Mattsson, J.; Håkansson, B. Seatrack Web: The HELCOM tool for oil spill prediction and identification of illegal polluters. In *Oil Pollution in the Baltic Sea*; The Handbook of Environmental Chemistry; Springer: Berlin/Heidelberg, Germany, 2011; Volume 27, pp. 155–183.
25. Aksamit, N.O.; Sapsis, T.; Haller, G. Machine-Learning Mesoscale and Submesoscale Surface Dynamics from Lagrangian Ocean Drifter Trajectories. *J. Phys. Oceanogr.* **2020**, *50*, 1179–1196. [CrossRef]
26. Haller, G.; Sapsis, T. Where do inertial particles go in fluid flows? *Phys. D Nonlinear Phenom.* **2008**, *237*, 573–583. [CrossRef]
27. Chorin, A.J. Numerical Solution of the Navier-Stokes Equations. *Math. Comput.* **1968**, *22*, 745–762. [CrossRef]
28. Price, K.; Storn, R.M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2006.
29. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995.
30. Hansen, N.; Müller, S.D.; Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **2003**, *11*, 1–18. [CrossRef]
31. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A Training Algorithm for Optimal Margin Classifiers. In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, New York, NY, USA, 27–29 July 1992; pp. 144–152.
32. Drucker, H.; Burges, C.J.; Kaufman, L.; Smola, A.J.; Vapnik, V. Support vector regression machines. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 1–6 December 1997.
33. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992.
34. Moody, J.; Darken, C.J. Fast Learning in networks of locally-tuned processing units. *Neural Comput.* **1989**, *1*, 281–294. [CrossRef]
35. Nair, V.; Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel, 21–24 June 2010; pp. 807–814.
36. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
37. Hochreiter, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **1998**, *6*, 107–116. [CrossRef]
38. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]

39. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
40. Frank, E.; Hall, M.; Trigg, L. *Weka 3: Data Mining Software in Java*; The University of Waikato: Hamilton, New Zealand, 2006.
41. Ketkar, N. Introduction to PyTorch. In *Deep Learning with Python*; Apress: Berkeley, CA, USA, 2017; pp. 195–208.
42. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
43. Golik, P.; Doetsch, P.; Ney, H. Cross-entropy vs. squared error training: A theoretical and experimental comparison. In Proceedings of the 14th Annual Conference of the International Speech Communication Association (Interspeech 2013), Lyon, France, 25–29 August 2013; pp. 1756–1760.
44. Seo, J.-H.; Lee, Y.H.; Kim, Y.-H. Feature Selection for Very Short-Term Heavy Rainfall Prediction Using Evolutionary Computation. *Adv. Meteorol.* **2014**, *2014*, 1–15. [CrossRef]
45. Kim, Y.-H.; Moon, S.-H.; Yoon, Y. Detection of Precipitation and Fog Using Machine Learning on Backscatter Data from Lidar Ceilometer. *Appl. Sci.* **2020**, *10*, 6452. [CrossRef]
46. Moon, S.-H.; Kim, Y.-H. Forecasting lightning around the Korean Peninsula by postprocessing ECMWF data using SVMs and undersampling. *Atmos. Res.* **2020**, *243*, 105026. [CrossRef]
47. Moon, S.-H.; Kim, Y.-H. An improved forecast of precipitation type using correlation-based feature selection and multinomial logistic regression. *Atmos. Res.* **2020**, *240*, 104928. [CrossRef]
48. Moon, S.-H.; Kim, Y.-H.; Lee, Y.H.; Moon, B.-R. Application of machine learning to an early warning system for very short-term heavy rainfall. *J. Hydrol.* **2019**, *568*, 1042–1054. [CrossRef]
49. Kim, H.-J.; Park, S.M.; Choi, B.J.; Moon, S.-H.; Kim, Y.-H. Spatiotemporal Approaches for Quality Control and Error Correction of Atmospheric Data through Machine Learning. *Comput. Intell. Neurosci.* **2020**, *2020*, 1–12. [CrossRef]
50. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

applied sciences

MDPI

Article

Salespeople Performance Evaluation with Predictive Analytics in B2B

Nelito Calixto [1] and João Ferreira [1,2,*]

[1] DCTI, Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR-IUL, 1649-026 Lisboa, Portugal; Nelito_Calixto@iscte-iul.pt

[2] INOV INESC Inovação—Instituto de Novas Tecnologias, 1000-029 Lisboa, Portugal

* Correspondence: jcafa@iscte-iul.pt

Received: 13 April 2020; Accepted: 8 June 2020; Published: 11 June 2020

Abstract: Performance Evaluation is a process that occurs multiple times per year on a company. During this process, the manager and the salesperson evaluate how the salesperson performed on numerous Key Performance Indicators (KPIs). To prepare the evaluation meeting, managers have to gather data from Customer Relationship Management System, Financial Systems, Excel files, among others, leading to a very time-consuming process. The result of the Performance Evaluation is a classification followed by actions to improve the performance where it is needed. Nowadays, through predictive analytics technologies, it is possible to make classifications based on data. In this work, the authors applied a Naive Bayes model over a dataset that is composed by sales from 594 salespeople along 3 years from a global freight forwarding company, to classify salespeople into pre-defined categories provided by the business. The classification is done in 3 classes, being: Not Performing, Good, and Outstanding. The classification was achieved based on KPI's like growth volume and percentage, sales variability along the year, opportunities created, customer base line, target achievement among others. The authors assessed the performance of the model with a confusion matrix and other techniques like True Positives, True Negatives, and F1 score. The results showed an accuracy of 92.50% for the whole model.

Keywords: data mining; predictive analytics; sales; performance measurement; human resources

1. Introduction

Salesperson performance measurement is a process that occurs multiple times per year on a company. The performance evaluation is based on various Key Performance Indicators (KPI's) extracted from multiple systems like Customer Relationship Management (CRM), and Enterprise Resource Planning (ERP).

Evaluating these KPI's can be time-consuming as they require the analysis of figures with complex calculations, a judgment based on the values, and the weight that each of the KPI's contributes to the performance as a whole. The KPI's often include the amount of products/services sold by the salesperson, the number of opportunities created, the ability to sell multiple products/services, the variability of the sales along the year, among many others. When a company has dozens or hundreds of salespeople, this process transforms on a thorough process that may involve other departments like Human Resources (HR) and Operations.

The result of the performance evaluation is a classification followed by actions to improve the performance where it is needed. Technology, through Data Mining (DM), currently is capable of make classification based on data. DM is the process of exploration and analysis, by automatic or semiautomatic means, of large quantities of data to discover meaningful patterns and rules [1]. DM tasks are classified into two categories: descriptive and predictive [1]. The predictive tasks, are the ones that perform inferences based on data to make predictions. The goal of these tasks is to create a predictive model. The goal of the predictive model is to allow the data miner to predict an unknown value of a specific variable. When the result of the prediction is a number, it is called a regression, and when the result is a label it's called a classification [1].

DM classification capabilities can help improving the process of the salesperson performance measurement. Companies can take advantage of the Predictive Analytics (PA) classification capabilities, to help on the judgment of KPI's that are based on complex calculations and the weight that each KPI contributes to the whole performance evaluation. By using classifications previously made by humans, companies can build models that can classify current sales of a salesperson and use them on the performance evaluation. Through these models it is possible to automate part of the performance evaluation process. The gains these automated evaluations can bring to the companies are among others:

- Reduction on the number of hours needed to analyse multiple KPI's to make a judgement of the salesperson performance, allowing the managers and salesperson to focus on other tasks that bring value to the business
- Improve the Salesperson Performance Appraisal process, by providing in advance, the possible future evaluation of the performance based on the salesperson current sales
- Allow the salesperson to act sooner in the performance measurement and appraisal time
- By allowing the salesperson to act sooner, companies can face reductions in salespeople turnover, and consequently reductions of costs on recruiting and training

In this work, the authors propose the use of DM techniques, to allow salesperson and sales leaders to make a better decision about salespeople performance measurement, by building a model in R that can classify a salesperson's performance based on metrics defined by the business. As many companies can have different evaluation processes, all companies in B2B area that has teams of salespeople being measured based on metrics, can take advantage of this DM process.

The dataset used for this analysis is composed of data regarding salespeople performance measurement from a Freight Forwarding global company. The sales are made by 594 salespeople between January 2017 and June 2019. This measurement is based on the company's internal performance measurement process, that are explained in this article on a very high level, to provide to the reader an understanding of the data, and the fields necessary to make the performance measurement. It is not the goal of this work to evaluate scientifically the process of salespeople performance measurement of this company.

1.1. Research Contribution

The DM process applied on this work can be replicated to any company who have historical objective metrics, and classifications applied to people based on these metrics.

The contributions of this paper are the followings:

1. Evaluate the use of predictive analytics process to classify salespeople
2. A novel form to use predictive analytics in the salesperson performance evaluation process
3. The use of predictive analytics to reduce the workload needed to prepare the performance evaluation of a salesperson
4. Automation in the analysis of several sales KPI's (objective measures) to get a classification of the responsible salesperson

1.2. Paper Structure

The paper is structured in the following way: Section 2 has the literature review; Section 3 has the background where the company's performance evaluation process is explained; Section 4 has the work methodology and all the steps needed to prepare the data for modeling and evaluation; Section 5 has the discussion; Section 6 has the conclusion, and proposals of future work.

2. Literature Review

2.1. Salesperson Performance

Academic studies demonstrate that the success of a salesperson normally has a direct relationship with the company performance, some authors states that: "When salespeople do well, the organization is likely doing well, and the contrary is normally true as well." [2]. When measuring salesperson performance, there are objective data, such as total sales increase, sales commissions or percent of quota, and subjective measures like manager's or peer's assessment of the salesperson [3]. Many companies use a combination of objective and subjective KPI's to make the assessment. A meta-analysis of objective and subjective sales indicators suggests that there is a low correlation identified between objective and subjective sales success indicators, which show that these indicators are not necessarily interchangeable, and the choice of the most appropriate may require trade-off [2].

The evaluation process of performance varies from company to company [4]. Activities on a job cannot be measured by only one method of objective or subjective measures, as some tasks of a job requires objective method of evaluation, and for others subjective measures are better. Bikrant Kesari examined the impact of objective and subjective measures of evaluation in sales departments, using various methods. For the company being studied, Bikrant Kesari concluded that objective measures were the most relevant factor used in the salesperson evaluation demonstrating the positive impact of the performance [4].

Muhammad Ruhul Amin et al., evaluated the effectiveness of weighted checklist method to appraise the performance of employees on different levels of a bank, based on Self assessment, Competency & demonstration of leadership behaviours, and Skill & knowledge assessment, the achievement classifications were made in 6 levels. The authors of the paper in question concluded that the impact of the method on employees was inevitable and all the financial and non financial benefits were effected due to the method [5].

John P. Campbell et al. defined individual job performance as things people do, and actions people take, that contribute to the organizations goals [6]. In another article Campbell et al. mention that performance is what facilitates achieving the organization goals directly [7].

The performance itself can be measured with judgmental and nonjudgmental measures which are the outcome measures [8]. The outcome measures use objective data, which don't need abstraction from who is collecting the data [9]. There are three predominant methods of measuring the sales performance. These are Outcome measure that are composed by sales volume and its variants, Judgmental managerial ratings and the salesperson Self-evaluation [10]. In the current work only objective measures are available, as the data provided for the current study only contain volume figures among other information related to sales, but none of these are related to subjective measures.

2.2. Predictive Analytics for Sales

Predictive analytics is an area increasingly entering the business and academic fields [11]. Companies more and more have been using DM to improve their internal processes and automate not only repetitive, but complex tasks nowadays completed by humans [12,13].

Authors in the academic area refer that PA has been used for several years by companies to get a competitive advantage, [14,15]. At first, by companies acting in the B2C with a large customer base and capacity to collect and store transactional data from customers, and only then by companies acting in the B2B area [14].

B2B selling companies are hiring cloud-based PA providers to draw on both inside and outside data sources to identify new leads so that they can take advantage of PA [16].

Mirzaei and Iyer did a comprehensive study on the application of PA over CRM data in 2014. The results show 57 articles found in 4 databases, where the studies focused on dimensions like Customer Acquisition, Attraction, Retention, Development, and Equity Growth [17]. Another fact the results show is that PA techniques between 2003 and 2013 gained a lot of popularity in areas like casinos, retailers, telecommunications, manufacturing, insurance and healthcare [17].

To understand what has been studied in the academic area in terms of predictive analytics, the authors hereunder describes some success cases of PA applied in sales forecasting.

2.2.1. Sales Forecasting of Computer Products Based on Variable Selection Scheme and Support Vector Regression (SVR)

Like many other industries, sales forecasting is also a challenge for computer product retailers. Wrong forecasts can cause product backlog or inventory shortages, incorrect customer demands and decrease customer satisfaction [18].

Chi-Jie Lu et al. combined Multi Variable Adaptive Regression Spines (MARS) with SVR to make a sales forecasting model for computer products. The main idea over the scheme was first to use MARS to select the essential forecasting variables and then use the identified key forecasting variables as the input variables for SVR. The data used was a compilation of the weekly sales data of five computer products from a computer retailer in Taiwan. The sales in the dataset referred to products like Notebooks, LCDs, Main Board, Hardrives, and Display cards [18].

2.2.2. Fast Fashion Sales Forecasting with Limited Data and Time

Another case of success found is applied to fast fashion, which is an industrial practice, where the main idea is to offer a continuous stream of new merchandise to the market [19]. With this practice, some fashion companies are even capable of having the products from the conceptual design to the final product in just two weeks. Companies working with this practice have to make their inventory decisions based on a forecast with short lead time and a tight schedule. The result is companies making a forecast on a near real-time basis and with a minimal amount of data. TM Choi et al. proposed an algorithm called Fast Fashion Forecasting (3F), that give the companies the ability to make forecasts with limited data and time. This algorithm uses two artificial intelligence methods: Extreme Learning Machine (ELM) and the Grey Model (GM). The data used belonged to a knitwear fashion company using a fast-fashion concept. The algorithm was tested with real and artificial sales data, and the results revealed an acceptable forecasting accuracy [19].

2.2.3. Support Vector Regression for Newspaper/Magazine Sales Forecasting

The next case is in the media area, where due to the constant transformations that information technologies are bringing to the world, new generations are more and more used to browse the internet for news and exciting stories [20]. With that in mind, the media industry also has to evolve to keep up with the progress. For that reason, it is more urgent for traditional media companies to make an accurate forecast on printing newspapers and magazines, to avoid excessive printing or not meeting the expected demand [20]. The authors of the study in question used SVR in a media company with printed newspaper/magazines to create a sales forecast that estimate and prepares the prints plan and distribution. The results of the study showed that SVR is a superior method in forecasting sales for the news/magazines industry [20].

With these scientific articles about success cases of PA in the B2C, we move next to success cases in the B2B area.

2.2.4. On Machine Learning towards Predictive Sales Pipeline Analytics

On companies operating in B2B, new sales are often identified as Leads. These leads move then into the Sales Opportunity Pipeline Management System. Later on, some of these Leads are qualified into opportunities. A sales opportunity is a set of one, or several products or services that the salesperson is trying to convert into a purchase. All the Opportunities are tracked, ideally ending on a won business that generates revenue for the company [21].

A fundamental part of the pipeline quality assessment is the lead-level win-propensity score identified as the win-propensity. The salesperson usually enters these scores, but to avoid noise inserted by the salesperson for various reasons and biased scores, the authors of the article in question proposed and successfully deployed a model to calculate the win-propensity using the Hawkes process model in a multinational Fortune 500 B2B-selling company in 2013 [21].

2.2.5. Prescriptive Analytics for Allocating Sales Teams to Opportunities

Still, in the Opportunities, other authors used Predictive and Prescriptive Analytics to increase the revenue of a company by 15%. Such increase was achieved by automating the allocation of sales resources to opportunities, to maximize opportunities revenue in B2B selling for the company [13].

The Predictive part was achieved by mining the historical selling data to learn sales response functions that have the behavioral relationship between the size and composition of a sales team, the revenue earned for the different types of customers, and the opportunities, through multiple linear regression [13].

For Prescriptive, these authors used the sales response functions to determine the allocation of salespeople's effort to the customer's opportunities that maximize the overall revenue earned by the salespeople, using a piece-wise linear approximation [13].

As presented in above articles, PA is widely being used on sales, the data used for these predictions is the data type needed to use in measurement of salespeople. With this base on PA for sales, the authors now moves to the application of PA in HR. HR is essential in this work due to the performance evaluation processes.

2.3. Predictive Analytics in HR Management

The articles studied in HR, refers to first how PA is being used for HR in general and then how PA is being used for people performance evaluation and analysis.

2.3.1. How PA Is Being Used for HR in General

An article published in 2017 [22], propose the use of PA in HR for:

- Employee Profiting and Segmentation, the authors propose that it can be achieved by anticipating the standing of every employee to profit from learning opportunities or capitalize on new undertakings;
- Employee Attrition and Loyalty Analysis, using predictive risk models to predict potential loss of employee, and by combining attrition risk score with worker performance info, HR can distinguish high-performing employees and also reduce potential attrition;
- Forecasting of HR Capacity and Recruitment Needs, using PA to anticipate the recruiting needs by combining the gap between people to recruit and people already employed, allowing HR to avoid under and over employment;

The authors of the article in question also proposes research in Appropriate Recruitment Profile Selection, Employee Sentiment Analysis, and Employee Fraud Risk management [22].

Sujeet N. Mishra et al. proposes the use of Human Resource Predictive Analytics (HRPA) for decision making by presenting two cases of success: One in a US wind turbine maker that changed the recruitment and retaining policies based on HRPA; Another is at Cisco, which used IBM SPSS to transform the relationship between its HR analysis and executive leaders [23]. Kessler et al. presents the categorization module of E-Gen, a modular system to treat job listings automatically. Through SVM these authors managed to rank candidate responses based on several information [24]. On another article, two authors used machine learning techniques to rank candidates on a recruiting process by analyzing the candidate adaptability to a job position based on the candidate tweets [25]. Other authors proposed an approach to evaluate job applications in online recruitment systems so they could solve the candidate ranking issue. They achieved this by analyzing the candidate's Linkedin profile and infer their personality characteristics using linguistic analysis on the candidate blog profile. For that, they had to use training data provided by human recruiters and applied in a large-scale recruitment scenario with three different positions and 100 applicants using Regression Tree and SVR [26].

2.3.2. How PA Is Being Used in HR for Performance Evaluation and Analysis

Zhao in his Conference Proceeding "International Seminar on Future Information Technology and Management Engineering" published in 2008, proposed a method of DM for performance evaluation. For that they gathered information about Ability, Attitude, Performance, Harvest, and Spirit in a dataset. Then they used the K-Expectation algorithm to classify employees into the same group. After that, a Decision tree is used to train a model based on rules that can be used by managers to classify and select the best employees from the applicants [27].

Jing applied Fuzzy Data Mining Algorithm (FDMA) for performance evaluation of human resources. For that, the author used evaluation records with four features: innovation ability, learning level, work efficiency, independence and workability, and each of these had four levels, which are the corresponding score of each feature [28]. Then, Jing used the maximal tree to cluster the human resource leading to the next step, that was to compare the data from management with each cluster and calculate the proximal values based on the FDMA, the last step referred to determine the evaluation. The evaluation, in this case, was a result closer to each of the 4 clusters that are named as Best, Better, General, and Worse [28].

Two authors applied Decision Trees on performance analysis of human resources to make classification analysis. The results show that there are mutual restraint and influence between performance results and working quality, tasks, skills, and attitude. Concluding that if the enterprise in the future cultivates employee working skills and quality, the employees will consciously improve themselves in these areas [29].

The above on PA for HR and performance evaluations are not based on data from sales made by salespeople. What is proposed in this article, is the use of PA to evaluate salesperson using the sales that was made by the salesperson, taking advantage of the data already available in the CRM, ERP systems, and previous performance evaluations. With that ground base, it is now time to proceed into the background section, where the company's salesperson evaluation process is described.

3. Background

In this section, the process and main KPI's used to evaluate the salesperson performance is described on a very high level, to provide an understanding of the data and fields used on this research. It is not the goal of this work to evaluate scientifically the process of salespeople performance measurement of this company.

3.1. Main KPI's Used for Salespeople Performance Evaluation

According to the process of the company that provided the data for this research, the main KPI's used to evaluate a salesperson performance are:

- Customer Base which is composed by the customers assigned to a salesperson in the current year
- Customer Base line that is the sum of volume sold to the Customer Base on the previous year (0 is assumed for new customers)
- Growth is the difference between the sum of the volume sold in the current year and the Base Line

3.2. Assess Salespeople Performance

Based on the company's performance evaluation process, there are a number of questions whose answer lead's to the evaluation level. The answer to these questions are provided by the KPI's described below:

- What growth did the salesperson brought to the company?

- The salesperson achieved the defined Targets?
- Do the assigned targets to the salesperson follow the company guidelines?
- Other relevant KPIs, on this stage, we will make a number of queries that goes from an in-depth analysis of the sales fluctuation and Customer Base, to the ratio of opportunities created for each customer

As displayed in Figure 1, the first level to verify is the growth, then check if the targets were achieved and finally if the targets follow the company guidelines. Other relevant KPIs that contribute to salesperson performance is also assessed, but these are the most important ones.

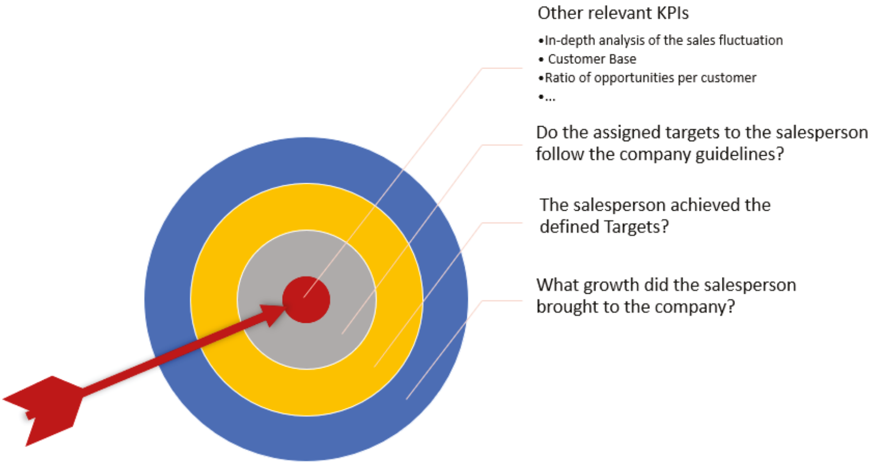

Other relevant KPIs
•In-depth analysis of the sales fluctuation
• Customer Base
•Ratio of opportunities per customer
•...
Do the assigned targets to the salesperson follow the company guidelines?
The salesperson achieved the defined Targets?
What growth did the salesperson brought to the company?

Figure 1. Company's performance evaluation stages.

3.2.1. What Growth Did the Salesperson Brought to the Company?

Starting with the first query: "What growth did the salesperson brought to the company?". A salesperson is assigned to an Account Base that has on average 70 customers, the base for analysis is the growth, which is the difference between the number of Twenty-foot equivalent unit (TEU) sold between the current and previous year. The base in the analysis is the sum of the growth for each year.

3.2.2. The Salesperson Achieved the Defined Targets?

The target definition in this company is supported on a top/down process. Targets are based on a roadmap that is defined globally by the sales controlling department, these targets are assigned for each region, and then distributed by the regional managers to the countries. The process continues until it reaches the salesperson. As exemplified in Figure 2 a global roadmap of 10,000 TEU's globally was defined. These TEU's are shared among all the regions, and ends on salesperson x and y in Lisbon with 30 TEU's each.

Although the company has implemented this process, not always the salesperson gets a reasonable target, because this will depend on the strategy defined by the local sales management, and on this company, part of the strategy is defined locally. For instance, in Figure 2, all Portugal's targets are assigned to Lisbon and none to Oporto. If the sales management in Portugal believe it's possible to achieve all targets with the 2 salespeople in Lisbon, they don't have to assign targets to salespeople in Oporto.

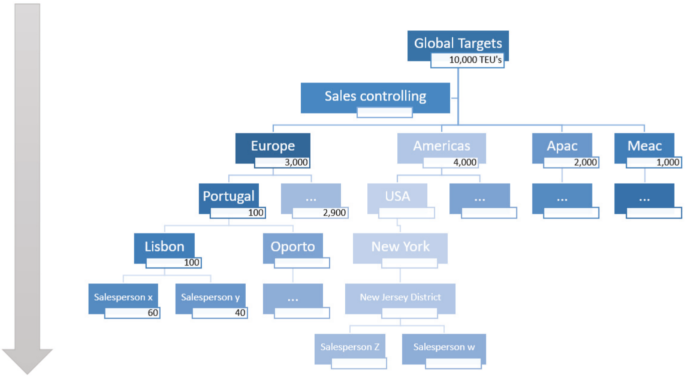


Figure 2. Target definition Top Down.

Other than the number of TEU's assigned for a region/country, there is also a target definition at the product level. This is another way of strategically redirect the sales team to target a specific product. For instance, if a country has a higher market for Import, the sales manager should set Targets on Import to boost Import sales.

3.2.3. Do the Assigned Targets to the Salesperson Follow the Company Guidelines?

In this company, targets are set to a salesperson based on 3 pillars:

- Account Base
- Sales roadmap
- Salesperson seniority

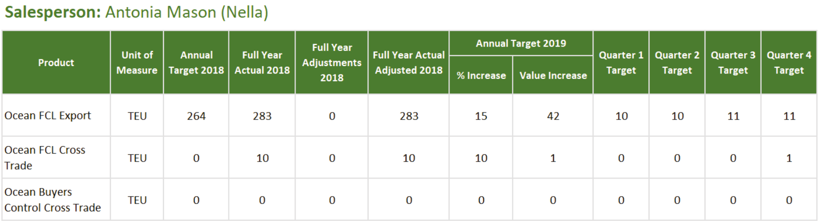
As described previously, the Account Base is composed of the customers that are assigned to the salesperson, and it has a significant impact on the level of the target that can be assigned to the person. If a salesperson has a Customer Base composed by 10 customers and these customers have a possibility of purchase 100 TEU's along the year, the targets assigned to this salesperson should not be a value that is too far from the 100 TEU's, unless the person who defines the targets have information's that may indicate that the customer will have a higher increase.

Sales roadmap is the document that has the plan for the company sales growth for the long term. This document for the company in question is composed of the main product categories, regions, trade lanes, among other information. Often sales managers set targets just based on the sales roadmap, but this may lead to the definition of "unrealistic" targets if the Account Base does not provide the potential needed to achieve the targets. When this happens, CRM Pipeline figures is another ally to set the targets. Usually, to improve target setting, Pipeline figures are added to the Sales Planning process. This way, the salesperson and manager have not only the Customer Base line but also the forecast (assuming good forecasting accuracy).

The salesperson seniority also has a significant role in how the salesperson works the Customer Base. A junior salesperson may not have the ability to manage complex accounts. Therefore the sales manager, when assigning the Customer Base, has to know the salesperson seniority. Seniority in the company/products has also consequences on managing the Account Base. For instance, if somebody has just joined the company and is also junior (young), he/she will need "more" time to start generating results: new company, new products, the need to build an internal network, among other relevant tasks. To mitigate this issue, often sales managers give a new/junior salesperson lower targets in the beginning and then increase the targets year-by-year as the seniority increases.

The Figure 3 displays an example of target definition for one salesperson (the name was replaced by one randomly generated for data protection), where it's possible to verify a 15% increase from the Account

Base line (Identified as the Full Year Actual Adjusted) that is 283 TEU's, the increase has an impact of 42 more TEU's, and is splitted across 4 quarters by 10 for Q1, 10 for Q2, 11 for Q3, and 11 for Q4.


Salesperson: Antonia Mason (Nella)

Product	Unit of Measure	Annual Target 2018	Full Year Actual 2018	Full Year Adjustments 2018	Full Year Actual Adjusted 2018	Annual Target 2019		Quarter 1 Target	Quarter 2 Target	Quarter 3 Target	Quarter 4 Target
						% Increase	Value Increase				
Ocean FCL Export	TEU	264	283	0	283	15	42	10	10	11	11
Ocean FCL Cross Trade	TEU	0	10	0	10	10	1	0	0	0	1
Ocean Buyers Control Cross Trade	TEU	0	0	0	0	0	0	0	0	0	0

Figure 3. Example of target setting in Incentive Management System.

Pipeline and seniority are entirely missing in this research, so to judge the targets, a validation is made comparing the targets directly with the Customer Base line in the dataset.

In this dataset, the evaluation is made by dividing the targets with the Customer Base line as displayed in the Formula (1).

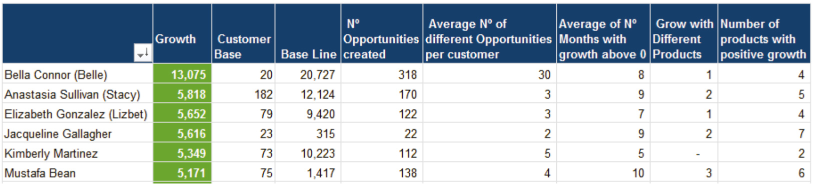
$$Target\ evaluation = \frac{Target}{AccountBaseline} \tag{1}$$

3.2.4. Other Relevant KPIs for the Salesperson Performance

There are other KPIs that need to be validated over the salesperson to measure the performance, these include:

- Customer Base, is the number of customers assigned to the salesperson
- Customer Base line, TEU's sold in the previous year to all the customers from the Customer Base
- Number of Opportunities created by the salesperson
- Average number of different Opportunities per customer
- Growth variability along the year (Number of months with positive growth)
- The salesperson ability to bring growth with Different Products
- The number of products with positive growth along the year

The table available in the Figure 4 provides all this information's for a sample of 5 salespeople. Worth of highlighting in the table is the number of opportunities of the first salesperson, which is remarkably high when compared to the second salesperson. Another important information is the average number of months with growth above 0, on average Bella Connor (Belle) is able to grow the Customer Base for about 8 months each year, and she can also grow more than one product.



	Growth	Customer Base	Base Line	Nº Opportunities created	Average Nº of different Opportunities per customer	Average of Nº Months with growth above 0	Grow with Different Products	Number of products with positive growth
Bella Connor (Belle)	13,075	20	20,727	318	30	8	1	4
Anastasia Sullivan (Stacy)	5,818	182	12,124	170	3	9	2	5
Elizabeth Gonzalez (Lizbet)	5,652	79	9,420	122	3	7	1	4
Jacqueline Gallagher	5,616	23	315	22	2	9	2	7
Kimberly Martinez	5,349	73	10,223	112	5	5	-	2
Mustafa Bean	5,171	75	1,417	138	4	10	3	6

Figure 4. Other relevant KPIs for 5 salespeople sorted by growth.

These rules generate a dataset of 42 KPI's, where based on the accumulated performance of the salesperson on each of the measures, a classification is possible to define for the salesperson. The classifications are divided into the following categories: Not Performing, Good, and Outstanding.

4. Work Methodology

The work methodology used in this research was the Cross Industry Standard Process for Data Mining (CRISP-DM). This methodology as presented in Figure 5 is divided into 6 stages. In this article, the authors describes the steps executed from stage 1 to 5, the last stage is not described here as requested by the company to not provide any information on that area.

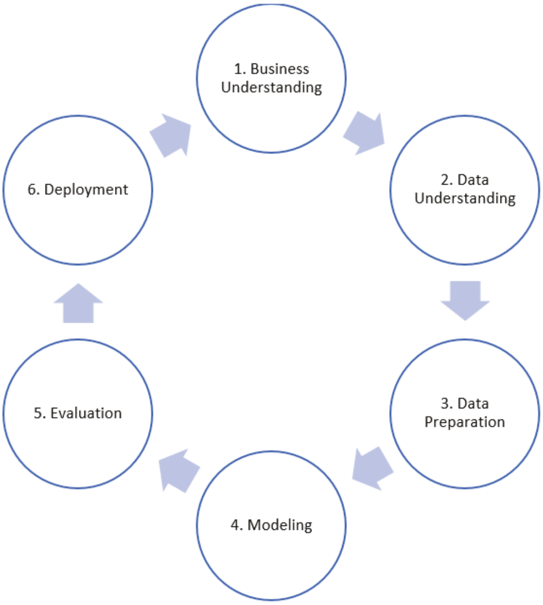


Figure 5. CRISP-DM Methodology adapted from [30].

The authors hereunder describes each of the CRISP-DM steps taken during this research following the CRISP-DM methodology.

4.1. Business Understanding

4.1.1. Objectives

With the main goal of classifying salespeople, and build a model that can tell if a salesperson is successful or not, this research project has the following business objectives:

- Identify the factors that contribute to the success of salespeople, based on the provided data
- Use predictive analytics process, to classify salespeople into 3 classes specified by the business, namely (Not Performing, Good, and Outstanding)

4.1.2. Business Success Criteria

The main success criteria for this research project is the ability to achieve the specific goals defined previously on the objectives. To evaluate these goals, the authors used the metrics provided by algorithms that measure the accuracy of the classifications.

4.2. Data Understanding

The data used in this research refers to sales between January 2017 and June 2019, from a freight forwarding company that operates worldwide on Air, Ocean, and Land. The sales were made by 594 salespeople. The data refers to shipments and sales opportunities for the customers grouped by year. As this company don't want to have their sensitive data provided to public, all sensitive data were removed from the dataset. Remaining only the figures and classification. The names of the salespeople were all replaced with names generated on a Name generator website [31].

There are 1071 rows and 45 columns. Each row represents all the sales, customer base, and sales opportunities made by one salesperson to all he/she's customer base along one year. The dataset has the following structure:

- Data reflects two and half years, respectively 2017, 2018, and part of 2019
- Data is grouped by, salesperson and year
- The volumes, base line, growth, target, and achievement are provided in separate columns for each of the six main products sold by the company, and one extra field for the remaining products growth percent grouped together
- Monthly variability is part of the dataset provided as number of months with positive growth
- The included Opportunity information refers only to win & implemented opportunities
- Target and achievements are included in volume and percentages
- The previous classifications used to train the model are defined into 3 classes (Not Performing, Good, and Outstanding)

Data Description

The dataset is publicly provided in the university online database. The data is provided on a csv file and the below tables (Tables 1 and 2) has the description of the attributes.

Table 1. Main attributes description.

Attribute Name	Description
Talent	The classification applied for the salesperson based on he's/she's performance for one year
Sales Person Code	Internal identification of the salesperson
Sales Person Name	Name of the salesperson
Year	Year the data refers to
Growth All Products	The Growth brought by the salesperson on that year, for all the products together (Growth calculation process is explained in Section 3.1)
Customer Base All Products	Number of customers assigned to the salesperson for each year
Base Line All Products	TEU's sold for all the Customer Base of the salesperson in the previous year for all the products
Growth Percent All Products	The result of the Growth All Products divided by the Base Line All Products
Target All Products	Sum of all the targets defined for each salesperson in one year, for all the products together
Target Achievement All Products	Target achievement for all the products, this is achieved by dividing the Target All Products by the Growth All Products, using the formula displayed in (2)
Number of Opportunities created	Count of opportunities won by the salesperson for each year
Average Opportunities per Customer	Average number of won opportunities per customer for each year
N^{o} Months with growth above 0	Number of months with growth above 0, for each year (for all the products together)
Number of Different Products with Growth	Count of the number of products that the salesperson can grow on one year
Growth with Different Products	Yes/No field identifying if the salesperson is able to grow more than one product on the year

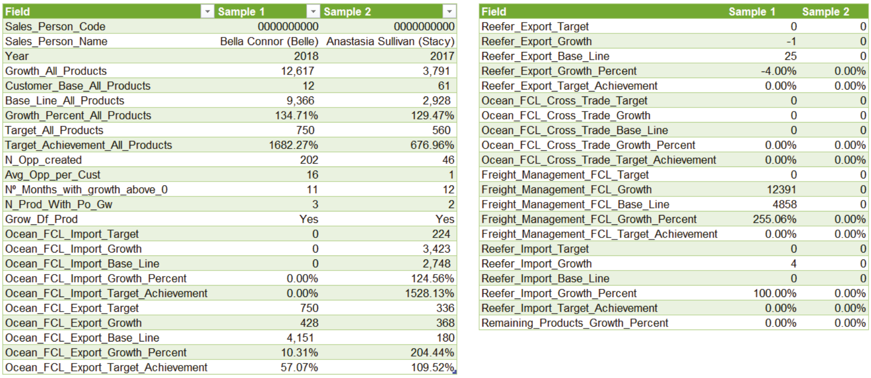
For each of the six main products, the following fields with performance indicators are also part of the dataset:

$$Target\ Achievement = \frac{Target}{Growth} \tag{2}$$

A sample of the dataset is provided on this work in the Figure 6 for better understanding.

Table 2. Product attributes description.

Attribute Name	Description
Target	The defined Target for the whole year
Growth	The sum of all the growth for the year
Base line	The sum of all the previous year TEU's for all the Customer Base of one year
Growth Percent	The result of the column Growth for the product in question divided by the Base line
Target Achievement	Calculation of the Target Achievement for the year, using the formula displayed in (2)



Field	Sample 1	Sample 2
Sales_Person_Code	0000000000	0000000000
Sales_Person_Name	Bella Connor (Belle)	Anastasia Sullivan (Stacy)
Year	2018	2017
Growth_All_Products	12,617	3,791
Customer_Base_All_Products	12	61
Base_Line_All_Products	9,366	2,928
Growth_Percent_All_Products	134.71%	129.47%
Target_All_Products	750	560
Target_Achievement_All_Products	1682.27%	676.96%
N_Opp_created	202	46
Avg_Opp_per_Cust	16	1
Nº_Months_with_growth_above_0	11	12
N_Prod_With_Po_Gw	3	2
Grow_Df_Prod	Yes	Yes
Ocean_FCL_Import_Target	0	224
Ocean_FCL_Import_Growth	0	3,423
Ocean_FCL_Import_Base_Line	0	2,748
Ocean_FCL_Import_Growth_Percent	0.00%	124.56%
Ocean_FCL_Import_Target_Achievement	0.00%	1528.13%
Ocean_FCL_Export_Target	750	336
Ocean_FCL_Export_Growth	428	368
Ocean_FCL_Export_Base_Line	4,151	180
Ocean_FCL_Export_Growth_Percent	10.31%	204.44%
Ocean_FCL_Export_Target_Achievement	57.07%	109.52%

Field	Sample 1	Sample 2
Reefer_Export_Target	0	0
Reefer_Export_Growth	-1	0
Reefer_Export_Base_Line	25	0
Reefer_Export_Growth_Percent	-4.00%	0.00%
Reefer_Export_Target_Achievement	0.00%	0.00%
Ocean_FCL_Cross_Trade_Target	0	0
Ocean_FCL_Cross_Trade_Growth	0	0
Ocean_FCL_Cross_Trade_Base_Line	0	0
Ocean_FCL_Cross_Trade_Growth_Percent	0.00%	0.00%
Ocean_FCL_Cross_Trade_Target_Achievement	0.00%	0.00%
Freight_Management_FCL_Target	0	0
Freight_Management_FCL_Growth	12391	0
Freight_Management_FCL_Base_Line	4858	0
Freight_Management_FCL_Growth_Percent	255.06%	0.00%
Freight_Management_FCL_Target_Achievement	0.00%	0.00%
Reefer_Import_Target	0	0
Reefer_Import_Growth	4	0
Reefer_Import_Base_Line	0	0
Reefer_Import_Growth_Percent	100.00%	0.00%
Reefer_Import_Target_Achievement	0.00%	0.00%
Remaining_Products_Growth_Percent	0.00%	0.00%

Figure 6. Sample of report data.

The classifications on the dataset, are made in the categories: Not Performing, Good, and Outstanding, these categories represents the following:

- Not Performing: as someone who has no growth, low or no Opportunities created, low target achievement and low growth over the months on one year
- Good: as someone who was able to grow the base line on at least 2 products, have positive growth for at least 7 months, have some opportunities, and a good target achievement
- Outstanding: as someone who was able to grow the base line on more than 2 products, or had an extremely high growth on one product, and have a positive growth along 8 months or more, have a good or high target achievement based on a large base line and high targets

4.3. Data Preparation

The dataset is composed of 45 columns and 1071 Rows. From the 45 columns, four have categorical data: these are Sales_Person_Code, Sales_Person_Name, Year, and Talent. The remaining columns have numerical data containing the salesperson's performance. A summary of the data available in the dataset is provided in the Table 3 for reference. The columns Sales_Person_Code, Sales_Person_Name, and Year were removed from the dataset, leaving the dataset with 42 columns.

Table 3. Table with classification statistics.

Field	Sample Value	Min	1st Quartile	Median	Mean	3rd Quartil	Max
Talent	Good	Not Performing: 373, Good: 269 and Outstanding: 53					
Growth_All_Products	−46	−1790	−26.5	37	138.4	205.5	12,617
Customer_Base_All_Products	4	1	7	14	15.97	22	83
Base_Line_All_Products	78	0	40.5	204	633.4	631.5	12,443
Growth_Percent_All_Products	−0.59	−1	0.17	0.18	3.17	1.08	566
Target_All_Products	0	0	120	272	405.7	560	6200
Target_Achievement_All_Products	0	−293	−0.09	0.11	−1.07	0.66	88
Nº_Opportunities_created	0	0	2	10	17.83	24	202
Average_Nº_of_Opportunities_per_customer	0	0	1	1	1.13	1	16
Nº_Months_with_growth_above_0	3	0	3	6	5.87	9	12
Nº_Different_Products	0	0	1	1	1.30	2	6
Grow_with_Different_Products	0	0	0	0	0.35	1	1
Ocean_FCL_Import_Target	0	0	0	101	220.90	300	2250
Ocean_FCL_Import_Growth	0	−2193	0	0	72.88	87	3423
Ocean_FCL_Import_Base_Line	0	0	0	37	303.80	233	12,199
Ocean_FCL_Import_Growth_Percent	0	−1	0	0	1.85	1	110,429
Ocean_FCL_Import_Target_Achievement	0	−185.50	0	0	0.03	0.41	88
Ocean_FCL_Export_Target	0	0	10	100	177.50	240	6200
Ocean_FCL_Export_Growth	−46	−1790	−7.5	0	35.07	56	3224
Ocean_FCL_Export_Base_Line	78	0	0	28	279.50	195.50	8413
Ocean_FCL_Export_Growth_Percent	−0.59	−1	−0.18	0	2.27	1	517.67
Ocean_FCL_Export_Target_Achievement	0	−293	−0.03	0	−0.99	0.33	27
Reefer_Export_Target	0	0	0	0	4.94	0	2000
Reefer_Export_Growth	0	−771	0	0	2.48	0	1923
Reefer_Export_Base_Line	0	0	0	0	16.41	0	5585
Reefer_Export_Growth_Percent	0	−1	0	0	0.10	0	46.8
Reefer_Export_Target_Achievement	0	−2	0	0	0.02	0	9.36
Ocean_FCL_Cross_Trade_Target	0	0	0	0	1.49	0	229
Ocean_FCL_Cross_Trade_Growth	0	−320	0	0	2.41	0	706
Ocean_FCL_Cross_Trade_Base_Line	0	0	0	0	4.07	0	1270
Ocean_FCL_Cross_Trade_Growth_Percent	0	−1	0	0	0.24	0	45.56
Ocean_FCL_Cross_Trade_Target_Achievement	0	−3.7	0	0	0	0	3.43
Freight_Management_FCL_Target	0	0	0	0	0.76	0	530
Freight_Management_FCL_Growth	0	−595	0	0	23.99	0	12,391
Freight_Management_FCL_Base_Line	0	0	0	0	24.07	0	4858
Freight_Management_FCL_Growth_Percent	0	−1	0	0	0.24	0	55.50
Freight_Management_FCL_Target_Achievement	0	0	0	0	0	0	0.51
Reefer_Import_Target	0	0	0	0	0.11	0	24
Reefer_Import_Growth	0	−186	0	0	0.91	0	172
Reefer_Import_Base_Line	0	0	0	0	3.11	0	501
Reefer_Import_Growth_Percent	0	−1	0	0	0.34	0	155
Reefer_Import_Target_Achievement	0	−0.65	0	0	0	0	0.58
Remaining_Products_Growth_Percent	0	−389	0	0	0.64	0	408

In the next sections, the authors submits the dataset to several techniques that evaluates the importance that each column may have to the model, and eliminates all the ones that contributes little or none. All the evaluations were made using RStudio, all the packages and functions used are identified.

The dataset contains:

- 695 rows classified by the business in a column called Talent
- 376 rows without classification, where the Talent column contains no data

In order to train the model, below evaluations and transformation were applied to the 695 classified rows.

The scripts used for this research are made in R language, using the free version of R Studio obtained from: [32] These scripts are provided in the university public database.

4.3.1. Near Zero Variance

Columns with low variance on the data, provide little or no knowledge to the models, so to improve the performance of the model, these columns can be eliminated. To Identify the columns that provide low knowledge, the authors used the function nearZeroVar from the carret package from R. This function diagnoses the predictors that have one unique value, or predictors that have few unique values in relative to the number of samples and the ratio of the frequency, from the most common value to the frequency of the second most common value.

From the results provided by the function, the most importants are zeroVar that has TRUE when the column contains only one distinct value and nzv, which has TRUE when the column in question has a near-zero variance predictor, for reference, the results are provided in the Table 4.

Table 4. Result of the nearZeroVar function.

Column	FreqRatio	PercentUnique	ZeroVar	nzv
Talent	1.39	0.43	FALSE	FALSE
Growth_All_Products	1.14	66.91	FALSE	FALSE
Customer_Base_All_Products	1.03	7.77	FALSE	FALSE
Base_Line_All_Products	3.91	64.89	FALSE	FALSE
Growth_Percent_All_Products	9.20	90.50	FALSE	FALSE
Target_All_Products	1.64	38.99	FALSE	FALSE
Target_Achievement_All_Products	14.67	89.93	FALSE	FALSE
N°_Opportunities_created	3.25	12.52	FALSE	FALSE
Average_N°_of_Opportunities_per_customer	3.83	1.29	FALSE	FALSE
N°_Months_with_growth_above_0	1.00	1.87	FALSE	FALSE
N°_Different_Products	1.97	1.01	FALSE	FALSE
Grow_with_Different_Products	1.87	0.29	FALSE	FALSE
Ocean_FCL_Import_Target	7.89	28.63	FALSE	FALSE
Ocean_FCL_Import_Growth	25.25	46.91	FALSE	FALSE
Ocean_FCL_Import_Base_Line	25.22	44.60	FALSE	FALSE
Ocean_FCL_Import_Growth_Percent	5.61	64.17	FALSE	FALSE
Ocean_FCL_Import_Target_Achievement	80.67	63.45	FALSE	FALSE
Ocean_FCL_Export_Target	5.23	28.78	FALSE	FALSE
Ocean_FCL_Export_Growth	12.83	44.75	FALSE	FALSE
Ocean_FCL_Export_Base_Line	14.07	41.29	FALSE	FALSE
Ocean_FCL_Export_Growth_Percent	2.41	62.88	FALSE	FALSE
Ocean_FCL_Export_Target_Achievement	48.50	65.61	FALSE	FALSE
Reefer_Export_Target	136.40	1.44	FALSE	TRUE
Reefer_Export_Growth	218.33	4.75	FALSE	TRUE

Table 4. *Cont.*

Column	FreqRatio	PercentUnique	ZeroVar	nzv
Reefer_Export_Base_Line	221.33	3.60	FALSE	TRUE
Reefer_Export_Growth_Percent	54.58	2.88	FALSE	TRUE
Reefer_Export_Target_Achievement	687.00	1.29	FALSE	TRUE
Ocean_FCL_Cross_Trade_Target	136.00	1.58	FALSE	TRUE
Ocean_FCL_Cross_Trade_Growth	214.00	5.47	FALSE	TRUE
Ocean_FCL_Cross_Trade_Base_Line	163.75	3.74	FALSE	TRUE
Ocean_FCL_Cross_Trade_Growth_Percent	49.38	4.60	FALSE	TRUE
Ocean_FCL_Cross_Trade_Target_Achievement	685.00	1.58	FALSE	TRUE
Freight_Management_FCL_Target	694.00	0.29	FALSE	TRUE
Freight_Management_FCL_Growth	25.33	10.94	FALSE	FALSE
Freight_Management_FCL_Base_Line	39.67	7.48	FALSE	TRUE
Freight_Management_FCL_Growth_Percent	7.82	8.20	FALSE	FALSE
Freight_Management_FCL_Target_Achievement	694.00	0.29	FALSE	TRUE
Reefer_Import_Target	344.50	0.72	FALSE	TRUE
Reefer_Import_Growth	39.00	4.17	FALSE	TRUE
Reefer_Import_Base_Line	64.80	3.45	FALSE	TRUE
Reefer_Import_Growth_Percent	20.13	4.32	FALSE	TRUE
Reefer_Import_Target_Achievement	691.00	0.72	FALSE	TRUE
Remaining_Products_Growth_Percent	42.86	5.61	FALSE	TRUE

There are 19 columns identified by the nearZeroVar function to be removed. After the removal of the 19 columns, the dataset still has 24 columns, 23 numerical + the Talent column.

4.3.2. Correlation Matrix

After the removal of the columns with low variance, a correlation matrix was applied to the remaining columns (excluding the Talent column), to find the ones that are highly correlated and remove at least one of them. For that, the authors used the function cor from the caret package. The cor function computes the variance, and the covariance of x and y. The results are a percentage of correlation between columns.

The result of the correlation matrix, as presented in the Figure 7, shows that there are 6 columns highly correlated (above 0.8). The authors eliminated three of the six columns, specifically: (Grow_with_Different_Products, Ocean_FCL_Export_Target_Achievement, and Ocean_FCL_Export_Growth_Percent). The dataset has now 21 columns, 20 numeric + the Talent. Only the columns with information specific to a product were removed, because between the columns referring to one product only and the overall, the overall provided more information to the dataset.

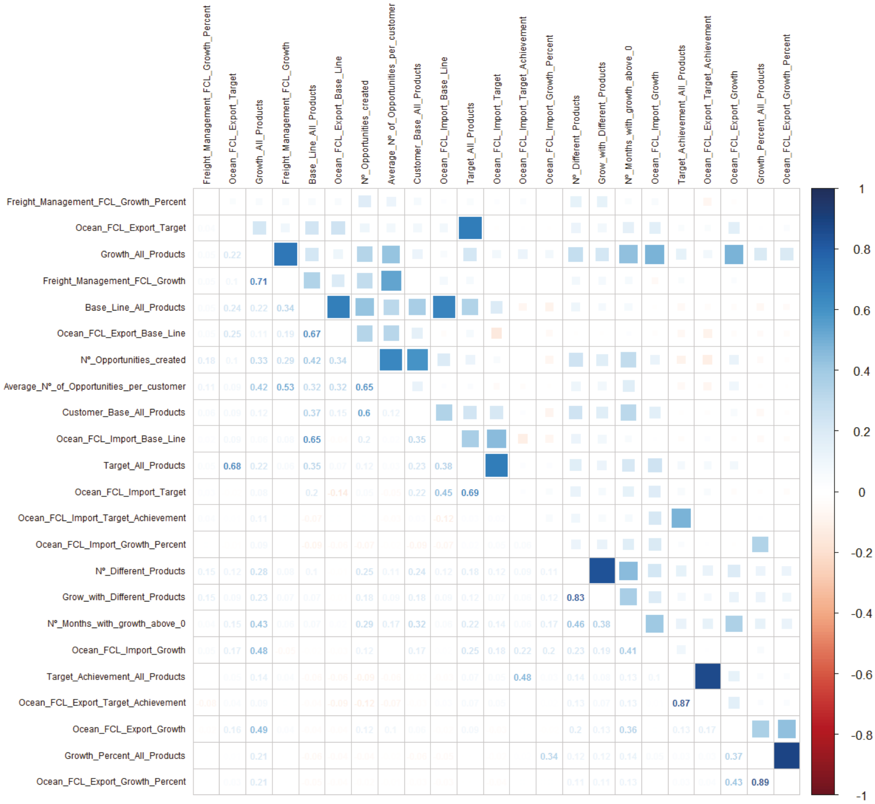


Figure 7. Correlation matrix.

4.3.3. Outliers Treatment

After removing the columns that contribute less, and the columns that are highly correlated, an outlier analysis to the remaining columns of the dataset was processed to identify them. Currently, there are 21 columns in the dataset, including the Talent column, which is the column with the classification.

The dataset has a high number of outliers, as it's possible to verify in the Figure 8. To identify the outliers, the authors used the boxplot.stats function of the package grDevices. This function is typically called by another function to build the boxplot. With that, it was possible to identify the outliers for all the 20 numeric columns.

To not remove data from the small dataset (695 rows from the training dataset), the outlier treatment was focused on applying to every outlier, the values in the range limit, obtained also using the boxplot.stats function from the package grDevices. The lower and higher values applied are provided in the Table 5 for reference, limits were applied to all columns except column: Nº_Months_with_growth_above_0 witch didn't needed.

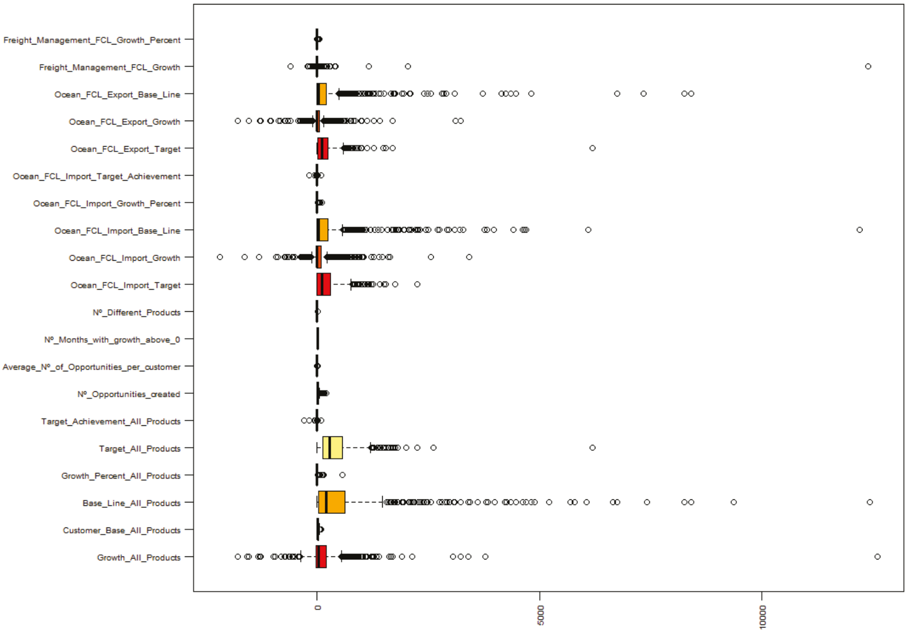


Figure 8. Outlier display.

Table 5. Outlier conversion table.

Field	Min Value	Max Value	Applied Lower-Value	Applied Higher Value
Growth_All_Products	−1790	12,617	−373	552
Customer_Base_All_Products	1	83	1	44
Base_Line_All_Products	0	12,443	0	1468
Growth_Percent_All_Products	−1	566	−1	2.94
Target_All_Products	0	6200	0	1200
Target_Achievement_All_Products	−293	88	−1.18	1.76
Nº_Opportunities_created	0	202	0	57
Average_Nº_of_Opportunities_per_customer	0	16	1	1
Nº_Months_with_growth_above_0	0	12	0	12
Nº_Different_Products	0	6	0	3
Ocean_FCL_Import_Target	0	2250	0	750
Ocean_FCL_Import_Growth	−2193	3423	−129	215
Ocean_FCL_Import_Base_Line	0	12,199	0	571
Ocean_FCL_Import_Growth_Percent	−1	110.43	−1	2.47
Ocean_FCL_Import_Target_Achievement	−185.50	88	−0.60	1.02
Ocean_FCL_Export_Target	0	6200	0	583
Ocean_FCL_Export_Growth	−1790	3224	−102	148
Ocean_FCL_Export_Base_Line	0	8413	0	482
Freight_Management_FCL_Growth	−595	12,391	0	0
Freight_Management_FCL_Growth_Percent	−1	55.50	0	0

After all the evaluations made, the authors discussed with the business the added value of the columns that refers to specific products, like Ocean FCL Export and Ocean FCL Import (the value added of Freight Management was practically removed by the fact that the outlier treatment eliminated all the values). The fact that these 2 products would be the only ones in the model would bias the salespeople that succeed more on these 2 products over the remaining products. Although the Overall Growth is still part of the dataset, the removal of all the columns specific for the products would produce similar results and with more value to the business. This lead to the removal of the other 10 columns. After the removal of these 10 columns, the dataset got reduced to 11 columns 10 numeric + 1 categorical.

4.3.4. Normalize Data

After the completion of all the data treatment steps, and as the Naive Bayes (NB) from R requires all the numeric columns to be standardized. The authors Standardized all the numeric columns using the function normalize of R from the BBmisc package.

With this task completed, the data treatment phase is concluded. The next phase is the evaluation where the results are assessed. This is described in the discussion section.

5. Discussion

5.1. Naive Bayes

In the research, from the studied algorithms, the authors selected the NB because of ease of it's implementation. The NB algorithm is a probabilistic classifier that selects each independent variable, and then associates it to a conditional probability. The conditional probability is calculated based on the following Formula (3)

$$P(C|A) = \frac{P(A|C) * P(C)}{P(A)} \tag{3}$$

The algorithm calculates the probability of an event occurs, based on another event that occurred in the past. For example, to predict if a salesperson may achieve his targets. In the formula, we can associate C to the probability of a salesperson achieving his targets, while A corresponds to the conditions that allowed the salesperson to achieve the targets, for instance, a customer base composed by customers that buy high volumes of TEU's.

The data was split into 2 separate datasets using the sample function in R, the training dataset with 70% of the data, which corresponds to 481 observations and the test dataset with 214 observations.

5.2. Identify Most Important Factors for Salesperson Success

To achieve the goal: Identify the most important factors for salesperson success, the authors built a Random Forest model with the same train dataset prepared for the NB model, but with the randomForest of R so that the function varImp could be used. The Random Forest model was created using the defaults of R, adding the following parameters: Type of random forest: classification, number of trees: 500, and No. of variables tried at each split: 2. The results were: Out of Bag (OOB) estimate of error rate: 2.91%, and the confusion matrix as provided in the Table 6.

Table 6. Confusion matrix from Random Forest.

	Good	Not Performing	Outstanding	Class.error
Good	183	0	8	0.04
Not Performing	1	250	0	0.00
Outstanding	7	0	32	0.18

The results of the varImp function are provided in the Table 7.

Table 7. Feature importance.

Feature	**Overall**
Growth_All_Products	132.94
Target_Achievement_All_Products	54.46
Growth_Percent_All_Products	26.35
N°_Months_with_growth_above_0	23.59
Target_All_Products	9.61
Base_line_All_Products	7.87
Customer_Base_All_Products	6.06
N°_Opportunities_created	4.63
N°_Different_Products	4.47
Average_N°_of_Opportunities_per_customer	0.22

The results show that the most important features are:

- Growth_All_Products
- Target_Achievement_All_Product
- Growth_Percent_All_Products
- N°_Months_with_growth_above_0

The remaining columns have residual importance compared to the ones before mentioned. The results go in line with the business people's opinions. The salesperson to succeed, have to: focus on growing the customer base, work to achieve their targets, and have steady positive growth for as many months as possible.

5.3. Run the Classification

The authors created a 20 Fold Cross Validation NB model based on the trainControl function from the carret package. Based on this model, the testing dataset was loaded and the predictions were requested.

A confusion matrix was built to evaluate the performance of the predictions made over the test dataset. The results are displayed in the Table 8.

The Accuracy (average) of the model is 92.52%. Based on the Confusion Matrix provided in the Table 8 it's possible to verify that the model only failed in 7.5% of the cases.

Table 8. Cross-Validated (20 fold) Confusion Matrix.

	Good	**Not Performing**	**Outstanding**
Good	35.80	2.70	0.80
Not Performing	2.30	49.50	0.0
Outstanding	1.70	0.00	7.30

An evaluation of the Precision, Specificity, Sensitivity, and an F1 score was made to evaluate the model accuracy and the results. As it's possible to verify in the Table 9, the Outstanding has a high Specificity but has a lower Sensitivity.

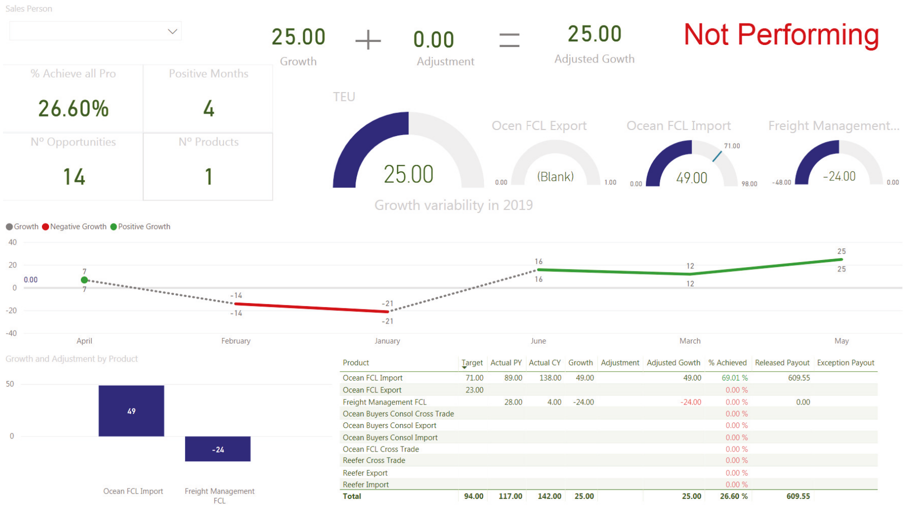
The F1 score display that the precision of the Not Performing is the highest, but for the Outstanding and Good classes, the accuracy of the tests made are high, which is very important considering that the results of this model are to evaluate people performance. Judging by the dataset size used on this analysis (695 observations), and analyzing it by the classes available, the Good has 269, Not Performing 373, and Outstanding 53. The scores obtained in the Detection Rates reflects the high number of correctly predicted evaluation for each class, and when compared to the Detection Prevalence it confirms the small number of erroneous predictions.

Table 9. Evaluation scores for NB model.

	Good	Not Performing	Outstanding
Sensitivity	85%	97%	67%
Specificity	93%	88%	100%
Pos Pred Value	87%	90%	100%
Neg Pred Value	91%	97%	97%
Precision	87%	90%	100%
Recall	85%	97%	67%
F1	86%	94%	80%
Prevalence	37%	53%	10%
Detection Rate	32%	51%	7%
Detection Prevalence	36%	57%	7%
Balanced Accuracy	88%	93%	83%

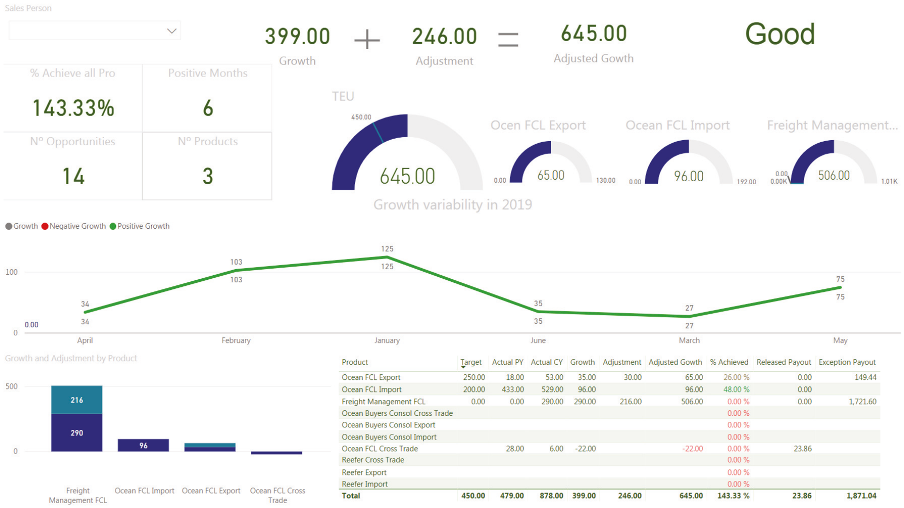
The limitation of this work was the data size and availability, as the number of observations available is not high and the number of observations between the available classes can differ. The authors believe that with a larger dataset, where it would be possible to extract data for each class with a similar number of observations, the model accuracy could be improved, and erroneous cases would decrease, leading to a more accurate model.

As the example, in the Figures 9–11, it's possible to review the results of the assessment in Power BI on a dashboard created for salesperson assessment, the dashboard has all the metrics and a classification made by the Predictive Analytics as Not Performing, Good and Outstanding, with this, all the objectives of the research are concluded successfully.



Product	Target	Actual PY	Actual CY	Growth	Adjustment	Adjusted Gowth	% Achieved	Released Payout	Exception Payout
Ocean FCL Import	71.00	89.00	138.00	49.00		49.00	69.01 %	609.55	
Ocean FCL Export	23.00						0.00 %		
Freight Management FCL		28.00	4.00	-24.00		-24.00	0.00 %	0.00	
Ocean Buyers Consol Cross Trade							0.00 %		
Ocean Buyers Consol Export							0.00 %		
Ocean Buyers Consol Import							0.00 %		
Ocean FCL Cross Trade							0.00 %		
Reefer Cross Trade							0.00 %		
Reefer Export							0.00 %		
Reefer Import							0.00 %		
Total	**94.00**	**117.00**	**142.00**	**25.00**		**25.00**	**26.60 %**	**609.55**	

Figure 9. Dashboard for a salesperson classified as Not Performing.



Sales Person

399.00 + 246.00 = 645.00 Good

Growth | Adjustment | Adjusted Gowth

% Achieve all Pro: 143.33%

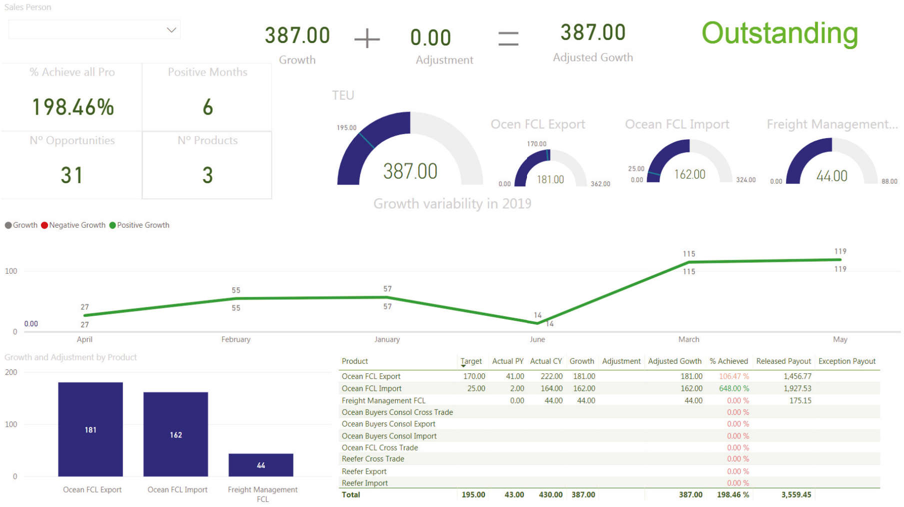
Positive Months: 6

Nº Opportunities: 14

Nº Products: 3

Product	Target	Actual PY	Actual CY	Growth	Adjustment	Adjusted Gowth	% Achieved	Released Payout	Exception Payout
Ocean FCL Export	250.00	18.00	53.00	35.00	30.00	65.00	26.00 %	0.00	149.44
Ocean FCL Import	200.00	433.00	529.00	96.00		96.00	48.00 %	0.00	
Freight Management FCL	0.00	0.00	290.00	290.00	216.00	506.00	0.00 %	0.00	1,721.60
Ocean Buyers Consol Cross Trade							0.00 %		
Ocean Buyers Consol Export							0.00 %		
Ocean Buyers Consol Import							0.00 %		
Ocean FCL Cross Trade		28.00	6.00	-22.00		-22.00	0.00 %	23.86	
Reefer Cross Trade							0.00 %		
Reefer Export							0.00 %		
Reefer Import							0.00 %		
Total	**450.00**	**479.00**	**878.00**	**399.00**	**246.00**	**645.00**	**143.33 %**	**23.86**	**1,871.04**

Figure 10. Dashboard for a salesperson classified as Good.



Sales Person

387.00 + 0.00 = 387.00 Outstanding

Growth | Adjustment | Adjusted Gowth

% Achieve all Pro: 198.46%

Positive Months: 6

Nº Opportunities: 31

Nº Products: 3

Product	Target	Actual PY	Actual CY	Growth	Adjustment	Adjusted Gowth	% Achieved	Released Payout	Exception Payout
Ocean FCL Export	170.00	41.00	222.00	181.00		181.00	106.47 %	1,456.77	
Ocean FCL Import	25.00	2.00	164.00	162.00		162.00	648.00 %	1,927.53	
Freight Management FCL		0.00	44.00	44.00		44.00	0.00 %	175.15	
Ocean Buyers Consol Cross Trade							0.00 %		
Ocean Buyers Consol Export							0.00 %		
Ocean Buyers Consol Import							0.00 %		
Ocean FCL Cross Trade							0.00 %		
Reefer Cross Trade							0.00 %		
Reefer Export							0.00 %		
Reefer Import							0.00 %		
Total	**195.00**	**43.00**	**430.00**	**387.00**		**387.00**	**198.46 %**	**3,559.45**	

Figure 11. Dashboard for a salesperson classified as Outstanding.

The steps presented above conclude the evaluation of the model performance. This was the last task in the research. In the next chapters, the authors concludes the research with a summary of the work and suggestions for future work.

6. Conclusions

In this work, the authors applied a Naive Bayes model to classify salespeople into pre-defined categories provided by the business. The classification is done in 3 classes, being: Not Performing, Good and Outstanding. The classification was achieved based on KPI's like growth volume and percentage, sales variability along the year, opportunities created, customer base line, target achievement among others.

The dataset is composed by 594 salespeople classified into three categories being these:

- Not Performing: as someone who has no growth, low or no Opportunities created, low target achievement and low growth over the months on one year
- Good: as someone who was able to grow the base line on at least 2 products, have positive growth for at least 7 months, have some opportunities, and a good target achievement
- Outstanding: as someone who was able to grow the base line on more than 2 products, or had an extremely high growth on one product, and have a positive growth along 8 months or more, have a good or high target achievement based on a large base line and high targets

The dataset used had in the beginning 45 columns. It was then reduced to 11 columns, based on several techniques to clean the data and evaluate the relevance of the columns to classify a salesperson's success. In this process, the authors also identified the most critical factors to evaluate a salesperson's performance based on the data, as Growth amount on all the products, Target achievement on all the products, Growth percentage on all the products, and the Number of Months with Growth above 0.

The model was evaluated with a confusion matrix and other techniques like True Positives, True Negatives, and F1 score. The results showed an Accuracy (average) of 92.52% for the whole model. For each of the classes in terms of precision, Not Performing has 90%, Good 87%, and Outstanding 100%. The F1 scores for Not Performing were 94%, for good 86%, and Outstanding 80%.

The accuracy results in this work are high because the size of the dataset and the variations of data have similar behavior for each of the classes. For instance, a salesperson not performing has in most of the time, low growth, low number of opportunities, and sales above 0 for a small number of months in one year; a good salesperson may have high growth in at least six months over one product; the outstanding salesperson should have growth extremely high for at least one product and growth above 0 for at least eight months.

This approach, when data is available, can help produce new guidelines that HR with pre-defined rules can use to automate part of the performance appraisal process. It can be applied to other cases and companies, and with DM, start automating the analysis of complex KPI's with relationships between them to generate a classification.

Future Work

As for future work, the authors proposes the use of a NB model to evaluate salespeople's performance with more CRM information. By taking advantage of other information that is also part of the salesperson job, information like the number Leads, activities (Visits, Calls), the other opportunity states, opportunities conversion rate, and the costs involved for each of the salespeople. The inclusion of subjective factors can also be part of the salesperson's performance. For instance, a more experienced salesperson may be training a junior salesperson, or taking several lost customers to recover, these facts can have an impact on the sales performance of the salesperson, the inclusion of flags that rate these can also be included.

All to aim towards a detailed and precise evaluation of salespeople's performance, increasing the fairness and reduce drastically the amount of work needed to make a performance evaluation for the salesperson.

Author Contributions: N.C. is a Master student that performed all development work. J.F. is a thesis supervisor and organized all work in the computer science subject. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by Portuguese National funds through FITEC programa Interface, with reference CIT "INOV—INESC Inovação—Financiamento Base".

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

B2C	Business to Consumer
B2B	Business to Business

CRM	Customer Relationship Management
MARS	Multi Variable Adaptive Regression Spines
SVR	Support Vector Regression
SVM	Support Vector Machines
3F	Fast Fashion Forecasting
ELM	Extreme Learning Machine
GM	Grey Model
HRPA	Human Resource Predictive Analytics
US	United States
FDMA	Fuzzy Data Mining Algorithm
OOB	Out of Bag
CRISP-DM	Cross Industry Standard Process for Data Mining
DM	Data Mining
KPI	Key Performance Indicator
ERP	Enterprise Resource Planning
NB	Naive Bayes
HR	Human Resources
TEU	Twenty-foot equivalent unit

References

1. Jain, N.; Srivastava, V. Data mining techniques: A survey paper. *Int. J. Res. Eng. Technol.* **2013**, *2*, 2319-1163.
2. Rich, G.A.; Bommer, W.H.; MacKenzie, S.B.; Podsakoff, P.M.; Johnson, J.L. Apples and apples or apples and oranges? A meta-analysis of objective and subjective measures of salesperson performance. *J. Pers. Sell. Sales Manag.* **1999**, *19*, 41–52.
3. Reday, P.A.; Marshall, R.; Parasuraman, A. An interdisciplinary approach to assessing the characteristics and sales potential of modern salespeople. *Ind. Mark. Manag.* **2009**, *38*, 838–844. [CrossRef]
4. Kesari, B. Salesperson Performance Evaluation: A Systematic Approach to Refining the Sales Force. *Int. J. Multidiscip. Manag. Stud.* **2014**, *4*, 49–66.
5. Amin, M.; Hossain, M.; Islam, M. Evaluating the Effectiveness of Weighted Checklist Method as a Tool of Employee Performance Appraisal: Evidence from Prime Bank Limited. *Stamford J. Bus. Stud.* **2015**, *6-II*, 32–47.
6. Campbell, J.P.; Wiernik, B.M. The Modeling and Assessment of Work Performance. *Annu. Rev. Organ. Psychol. Organ. Behav.* **2015**, *2*, 47–74. [CrossRef]
7. Campbell, J.P.; McCloy, R.A.; Oppler, S.H.; Sager, C.E. A theory of performance. *Pers. Sel. Organ.* **1993**, *3570*, 35–70.
8. Levy, M.; Sharma, A. Relationships among measures of retail salesperson performance. *J. Acad. Mark. Sci.* **1993**, *21*, 231–238. [CrossRef]
9. Landy, F.J.; Farr, J.L. *The Measurement of Work Performance: Methods, Theory, and Applications*; Academic Press: Cambridge, MA, USA, 1983.
10. Cannon, J.P.; Spiro, R. The Measurement of Salesperson Performance: Comparing Self-Evaluations with Customer Evaluations. In *Enhancing Knowledge Development in Marketing—1991 Summer Educators' Proceedings*; American Marketing Association: Chicago, IL, USA, 1991; pp. 1–10.
11. Waller, M.A.; Fawcett, S.E. Data science, predictive analytics, and big data: A revolution that will transform supply chain design and management. *J. Bus. Logist.* **2013**, *34*, 77–84. [CrossRef]
12. Ryu, S.; Siegel, E. *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie or Die Book Review*; Healthc Inform Research; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2013; Volume 19, pp. 63–65. [CrossRef]
13. Kawas, B.; Squillante, M.S.; Subramanian, D.; Varshney, K.R. Prescriptive Analytics for Allocating Sales Teams to Opportunities. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops, Dallas, TX, USA, 7–10 December 2013; pp. 211–218. [CrossRef]
14. Domingos, R.; Van de Merckt, T. Best Practices for Predictive Analytics in B2B Financial Services. In Proceedings of the 2010 Conference on Data Mining for Business Applications, August 2010; pp. 35–48. Available online: https://dl.acm.org/doi/10.5555/1893248.1893253 (accessed on 1 February 2020).

15. Bose, R. Advanced analytics: Opportunities and challenges. *Ind. Manag. Data Syst.* **2009**, *109*, 155–172. [CrossRef]
16. Lilien, G.L. The B2B Knowledge Gap. *Int. J. Res. Mark.* **2016**, *33*, 543–556. [CrossRef]
17. Mirzaei, T.; Iyer, L. Application of predictive analytics in customer relationship management: A literature review and classification. In Proceedings of the Southern Association for Information Systems Conference, Macon, GA, USA, 21–22 March 2014; pp. 1–7.
18. Lu, C.J. Sales forecasting of computer products based on variable selection scheme and support vector regression. *Neurocomputing* **2014**, *128*, 491–499. [CrossRef]
19. Choi, T.M.; Hui, C.L.; Liu, N.; Ng, S.F.; Yu, Y. Fast fashion sales forecasting with limited data and time. *Decis. Support Syst.* **2014**, *59*, 84–92. [CrossRef]
20. Yu, X.; Qi, Z.; Zhao, Y. *Support Vector Regression for Newspaper/Magazine Sales Forecasting*; Procedia Computer Science; Elsevier: Amsterdam, The Netherlandsm, 2013; Volume 17, pp. 1055–1062. [CrossRef]
21. Yan, J.; Zhang, C.; Zha, H.; Gong, M.; Sun, C.; Huang, J.; Chu, S.; Yang, X. On machine learning towards predictive sales pipeline analytics. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
22. Malisetty, S.; Archana, R.; Kumari, V. Predictive Analytics in HR Management. *Indian J. Public Health Res. Dev.* **2017**, *8*, 115. [CrossRef]
23. Mishra, S.N.; Lama, D.R.; Pal, Y. Human Resource Predictive Analytics (HRPA) For HR Management in Organizations. *Int. J. Sci. Technol. Res.* **2019**, *5*, 3.
24. Kessler, R.; Torres-Moreno, J.M.; El-Bèze, M. E-Gen: Automatic Job Offer Processing System for Human Resources. In *MICAI 2007: Advances in Artificial Intelligence*; Gelbukh, A., Kuri Morales, Á.F., Eds.; Springer: Berlin, Germany, 2007; pp. 985–995.
25. Menon, V.M.; Rahulnath, H.A. A novel approach to evaluate and rank candidates in a recruitment process by estimating emotional intelligence through social media data. In Proceedings of the 2016 International Conference on Next Generation Intelligent Systems (ICNGIS), Kottayam, India, 1–3 September 2016; pp. 1–6. [CrossRef]
26. Faliagka, E.; Ramantas, K.; Tsakalidis, A.; Tzimas, G. Application of machine learning algorithms to an online recruitment system. In Proceedings of the International Conference on Internet and Web Applications and Services, Stuttgart, Germany, 27 May–1 June 2012.
27. Zhao, X. A Study of Performance Evaluation of HRM: Based on Data Mining. In Proceedings of the 2008 International Seminar on Future Information Technology and Management Engineering, Leicestershire, UK, 20 November 2008; pp. 45–48. [CrossRef]
28. Jing, H. Application of Fuzzy Data Mining Algorithm in Performance Evaluation of Human Resource. In Proceedings of the 2009 International Forum on Computer Science-Technology and Applications, Chongqing, China, 25–27 December 2009; Volume 1, pp. 343–346. [CrossRef]
29. Xiaofan, C.; Fengbin, W. Application of Data Mining on Enterprise Human Resource Performance Management. In Proceedings of the 2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering, Kunming, China, 26–28 November 2010; Volume 2, pp. 151–153. [CrossRef]
30. Wikipedia. Cross-Industry Standard Process for Data Mining. 2020. Available online: https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining (accessed on 12 May 2020).
31. Masterpiece Generator. Name Generator. 2020. Available online: https://www.name-generator.org.uk/ (accessed on 1 February 2020).
32. RStudio.com. R Studio. 2019. Available online: https://rstudio.com/ (accessed on 22 October 2019).

MDPI
St. Alban-Anlage 66
4052 Basel
Switzerland
Tel. +41 61 683 77 34
Fax +41 61 302 89 18
www.mdpi.com

Applied Sciences Editorial Office
E-mail: applsci@mdpi.com
www.mdpi.com/journal/applsci

MDPI

Academic Open
Access Publishing

www.mdpi.com

ISBN 978-3-0365-7907-8