algorithms

Special Issue Reprint

Machine Learning for Pattern Recognition (2nd Edition)

Edited by
Chih-Lung Lin, Bor-Jiunn Hwang, Shaou-Gang Miaou and Chi-Hung Chuang

mdpi.com/journal/algorithms

MDPI

Machine Learning for Pattern Recognition (2nd Edition)

Machine Learning for Pattern Recognition (2nd Edition)

Guest Editors

Chih-Lung Lin
Bor-Jiunn Hwang
Shaou-Gang Miaou
Chi-Hung Chuang

MDPI

Basel • Beijing • Wuhan • Barcelona • Belgrade • Novi Sad • Cluj • Manchester

Guest Editors

Chih-Lung Lin
Department of Electrical Engineering
Hwa Hsia University of Technology
New Taipei City
Taiwan

Bor-Jiunn Hwang
Department of Applied Artificial Intelligence
Ming Chuan University
Taoyuan
Taiwan

Shaou-Gang Miaou
Department of Electronic Engineering
Chung Yuan Christian University
Taoyuan City
Taiwan

Chi-Hung Chuang
Department of Information and Computer Engineering
Chung Yuan Christian University
Taoyuan
Taiwan

Editorial Office

MDPI AG
Grosspeteranlage 5
4052 Basel, Switzerland

This is a reprint of the Special Issue, published open access by the journal *Algorithms* (ISSN 1999-4893), freely accessible at: https://www.mdpi.com/journal/algorithms/special_issues/P3JE499V45.

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

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

ISBN 978-3-7258-6350-1 (Hbk)
ISBN 978-3-7258-6351-8 (PDF)
https://doi.org/10.3390/books978-3-7258-6351-8

© 2026 by the authors. Articles in this book are Open Access and distributed under the Creative Commons Attribution (CC BY) license. The book as a whole is distributed by MDPI under the terms and conditions of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/).

Contents

Bedatri Moulik, Sanmukh Kaur and Muhammad Ijaz
Optimized Energy Consumption of Electric Vehicles with Driving Pattern Recognition for Real Driving Scenarios
Reprinted from: *Algorithms* **2025**, *18*, 204, https://doi.org/10.3390/a18040204 . . . . . **1**

Sadeem Alrasheed, Suliman Aladhadh and Abdulatif Alabdulatif
Protecting Intellectual Security Through Hate Speech Detection Using an Artificial Intelligence Approach
Reprinted from: *Algorithms* **2025**, *18*, 179, https://doi.org/10.3390/a18040179 . . . . . **18**

Roxana-Elena Tudoroiu, Mohammed Zaheeruddin, Daniel-Ioan Curiac, Mihai Sorin Radu and Nicolae Tudoroiu
Investigations into the Design and Implementation of Reinforcement Learning Using Deep Learning Neural Networks
Reprinted from: *Algorithms* **2025**, *18*, 170, https://doi.org/10.3390/a18030170 . . . . . **44**

Hamid Reza Aghamiri, S. Abolfazl Hosseini, James R. Green and B. John Oommen
Nonparametric Probability Density Function Estimation Using the Padé Approximation
Reprinted from: *Algorithms* **2025**, *18*, 88, https://doi.org/10.3390/a18020088 . . . . . **97**

Yao-San Lin, Hung-Yu Chen, Mei-Ling Huang and Tsung-Yu Hsieh
Data Augmentation for Voiceprint Recognition Using Generative Adversarial Networks
Reprinted from: *Algorithms* **2024**, *17*, 583, https://doi.org/10.3390/a17120583 . . . . . **119**

Pierre Gouton, Kacoutchy Jean Ayikpa and Diarra Mamadou
Estimating the Spectral Response of Eight-Band MSFA One-Shot Cameras Using Deep Learning
Reprinted from: *Algorithms* **2024**, *17*, 473, https://doi.org/10.3390/a17110473 . . . . . **140**

Markus Eisenbach, Andreas Gebhardt, Dustin Aganian and Horst-Michael Gross
Improving Re-Identification by Estimating and Utilizing Diverse Uncertainty Types for Embeddings
Reprinted from: *Algorithms* **2024**, *17*, 430, https://doi.org/10.3390/a17100430 . . . . . **160**

Sayed Mehedi Azim, Aravind Balasubramanyam, Sheikh Rabiul Islam, Jinglin Fu and Iman Dehzangi
Explainable Machine Learning Model to Accurately Predict Protein-Binding Peptides
Reprinted from: *Algorithms* **2024**, *17*, 409, https://doi.org/10.3390/a17090409 . . . . . **200**

Yufei Shen and Wenxing Zhou
Classification and Regression of Pinhole Corrosions on Pipelines Based on Magnetic Flux Leakage Signals Using Convolutional Neural Networks
Reprinted from: *Algorithms* **2024**, *17*, 347, https://doi.org/10.3390/a17080347 . . . . . **217**

Md. Mahfuzur Rahman, Sunzida Siddique, Marufa Kamal, Rakib Hossain Rifat and Kishor Datta Gupta
UAV (Unmanned Aerial Vehicle): Diverse Applications of UAV Datasets in Segmentation, Classification, Detection, and Tracking
Reprinted from: *Algorithms* **2024**, *17*, 594, https://doi.org/10.3390/a17120594 . . . . . **236**

algorithms

MDPI

Article

Optimized Energy Consumption of Electric Vehicles with Driving Pattern Recognition for Real Driving Scenarios

Bedatri Moulik [1], Sanmukh Kaur [2,*] and Muhammad Ijaz [3]

[1] Amity Institute of Technology, Amity University Uttar Pradesh, Noida 201301, India
[2] Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida 201313, India
[3] Department of Engineering, Manchester Metropolitan University, Manchester M1 5GD, UK
* Correspondence: skaur2@amity.edu or sanmukhkaur@gmail.com; Tel.: +91-9810549015

Abstract

Energy management strategies (EMS) in the context of electric or hybrid vehicles can optimize the available energy by minimizing consumption. Most optimization-based EMS are not real-time-applicable for an accurate estimation of future consumption. The performance of these strategies also strongly depends on the driving patterns, which may be influenced by road and traffic conditions, among other factors such as driving style, weather, vehicle type, etc. The primary contribution of this work is to develop a novel two-layer driving pattern recognition (DPR) system for roadway type and traffic classification, thus enabling the identification of unknown patterns for the enhancement of the prediction of energy consumption of an electric vehicle (EV). The novelty of this work lies in the development of a strategy based on real-time data which is capable of classifying driving patterns and implementing an optimized EMS based on the results of the DPR. In the approach, first, labels are defined based on statistical features related to speed followed by the creation of representative driving patterns (RDPs). A neural network-based classifier is then employed for classification into six classes based on four features. A training accuracy of 97.7% is achieved with the classification of unknown speed profiles into the known RDPs. Testing with patterns from two different test routes shows an accuracy of 97.45% and 96.98% during morning and 96.65% and 94.12% during evening hours, respectively. Apart from the route and time of data collection, accuracy is also a function of sampling time horizon and the threshold values chosen for the features. A sensitivity analysis was also performed to evaluate the relative importance of each feature. An EMS based on sequential quadratic programming (SQP) was combined with DPR for the computation of optimal energy consumption. Simulation results show that maximum and minimum energy savings of 61% and 18% were obtained under suburban low traffic and highway high traffic conditions, respectively. An eco-driving or driver speed advisory system may further be developed based on information obtained from multiple routes and varying traffic scenarios.

Keywords: electric vehicles; energy consumption; driving pattern recognition; representative driving cycles; optimization

1. Introduction

Electric and hybrid vehicles (HV) are becoming increasingly popular due to their environmental benefits. Energy management strategies (EMSs) include different methods or techniques that aim to manage multiple aspects of optimal energy consumption to

improve the performance of electric vehicles (EVs) [1]. The integration of EMS with EVs as a part of the eco-driving protocol may enable the smart management of driving speed profiles and help in achieving long-term sustainability goals. The main significance of optimized energy management lies in the achievement of better performance of vehicles with the desired set of characteristics, as defined by the user. An optimal EMS is an important requirement to be implemented to produce a final product which is not only smart and automated but also energy efficient.

In the past [2–6], different types of methods have been adopted for energy management in EVs or hybrid electric vehicles (HEVs). Depending on the mode of operation, the strategies can be classified as online or offline. In Figure 1, a broad classification of EMSs is listed based on online and offline approaches.

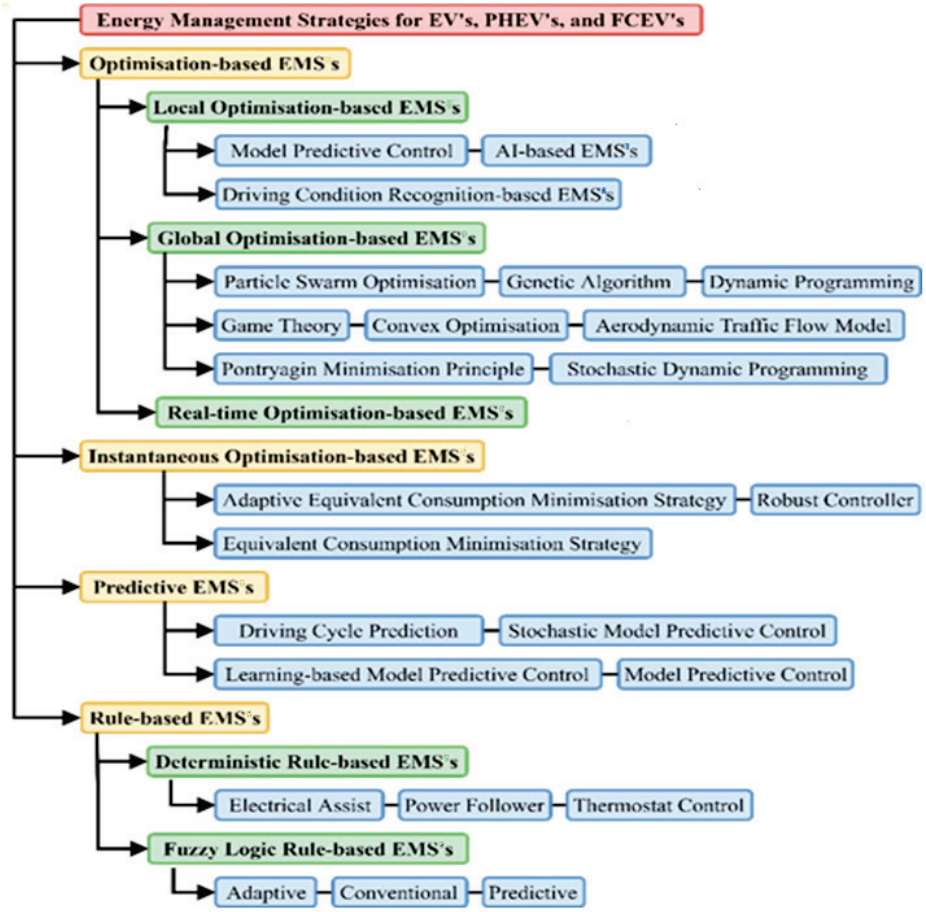


Figure 1. Classification of EMSs for electric and HVs [2–6].

Online approaches are generally real-time-based algorithms such as predictive and instantaneous optimization techniques, whereas offline approaches are non-real-time and can provide a globally optimal solution. Predictive (prediction-based) and cognitive (recognition-based) energy management can enhance the performance of electric or HVs immensely [7]. In [8], a dynamic programming-based energy management strategy (EMS) is presented along with a general regression neural network (NN) for improving the system efficiency of a dual-motor battery electric vehicle (BEV). In [9], authors have developed an energy management system for the minimization of the total distribution costs of BEV batteries, which is considered along with the maximization of average utilization of batteries. In [10], a plug-in hybrid vehicle has been optimized for better fuel efficiency by using an adaptive and hierarchical EMS and classification based on fuzzy logic control. In [11], an adaptive wavelet transform–fuzzy logic control EMS has been

combined with cluster analysis-based classification to divide power among the hybrid energy sources in an optimal manner. In [12], the fuel cell lifetime of HEVs is improved by using a NN classification that works with a genetic-algorithm-based EMS. Similarly, [13] achieved an improvement in the fuel efficiency of HEVs by optimizing power management using fuzzy logic-based strategies. In [14], the authors present a fuzzy-based EMS for fuel cell vehicles (FCV) by employing a particle swarm optimization algorithm for the better recognition of traffic conditions.

Further, in [15], a multi-mode EMS has been achieved for a fuel cell electric vehicle (FCEV). A Markov chain-based DPR classification has been employed for the improvement of fuel economy. In [16], authors minimized the fuel consumption of HEVs using the Euclid approach degree-based classification and simulated annealing-based EMS. In [17], the fuel consumption of HEVs has been minimized by using heuristic and optimal control strategies.

Similarly, in [18], fuel consumption and emission improvements of an HEV have been analyzed using clustering and torque division by employing classification and optimization, respectively. Further, in [19], a DPR-based optimization technique has been presented with a learning vector quantization classification system to improve the fuel economy of HEVs. In the case of an EV, a prior estimate of the load can be made depending upon the driving pattern of the driver, and information thus obtained can be used to extend the life and range of the battery [20]. As concluded from the literature, regardless of the type of vehicle, combining DPR with energy management may lead to significant improvements in fuel economy, energy consumption, battery life, performance, and efficiency [21,22]. Most of the existing literature focuses on the optimization of the energy consumption in electric vehicles; however, achieving real-time prediction and optimization of energy consumption is a challenging task, necessitating the need for the development of DPR-based EMS. DPR may enable driving suggestions and the identification of traffic congestion patterns along with suggestions of energy-efficient routes. It may also lead to the implementation of an optimized EMS to control the vehicle speed and realize eco-driving. Minimizing the energy consumption of an EV considering a given route and traffic situation is one of the goals of eco-driving. However, due to several technical, computational, and operational challenges, the realization of eco-driving becomes difficult. As decisions when driving must be made within milliseconds, real-time-applicable DPR and EMS are required.

In the present work, real-time-applicable EMS is integrated with DPR to overcome the challenges faced in achieving optimality in real-time scenarios. The novelty of the contribution lies in the development of the two-layer DPR, which is capable of organizing unknown driving patterns into known representative driving patterns (RDPs) by recognizing the patterns in terms of road and traffic conditions. Longer and shorter sampling horizons have been chosen for road and traffic conditions, respectively. RDPs are used to represent road and traffic conditions for a certain driving route. The characterization of real and standard driving patterns has have been performed in terms of four features, namely average speed, positive and negative acceleration, and number of stops. Unknown patterns are classified into the known RDPs by employing a multi-layer perceptron neural network (MLPNN) with six classes, namely highway high and low traffic, sub-urban high and low traffic, and urban high and low traffic. A sequential quadratic programming (SQP)-based energy management strategy (EMS) is considered for the optimization of energy consumption of an EV. The feasibility of DPR is verified by subjecting the classifier to an unknown mixed pattern generated as a result of combining different standard cycles. The DPR is also evaluated for two test route data points to evaluate the classification accuracy with changing routes and data collection timings. The impact of road and traffic on energy consumption prediction has been evaluated on the different test data. A comparison analysis of results has also been performed to indicate that an EMS with DPR can lead to more energy savings.

This paper is organized as follows. In Section 2, DPR is developed for roadway type and traffic recognition. Section 2.4 discusses EMS problem formulation using sequential quadratic programming (SQP). The results and discussion are presented in Section 3, followed by the conclusion in Section 4.

2. DPR for Roadway and Traffic Type Classification

In this section, a two-layer DPR is discussed. The first step is to prepare the data by sampling at different intervals, and labeling the sampled data. Three target classes are defined based on roadway type and three based on traffic. The main idea is to generate RDPs corresponding to roadways and traffic situations for a certain route. The generation of RDPs is based on data collected along the real route and information available from standard cycles. Finally, an MLPNN-based classifier is used to classify an unknown pattern into the target classes.

2.1. Data Analysis and Formation of RDPs

As a first step, data are collected using an on-board diagnostic (OBD) device from a fixed route at different times of the day for a month. The selected route is a drive over a distance of 10.4 km. The differences in morning and evening average travel times are noted due to variations in traffic, as shown in Figure 2.

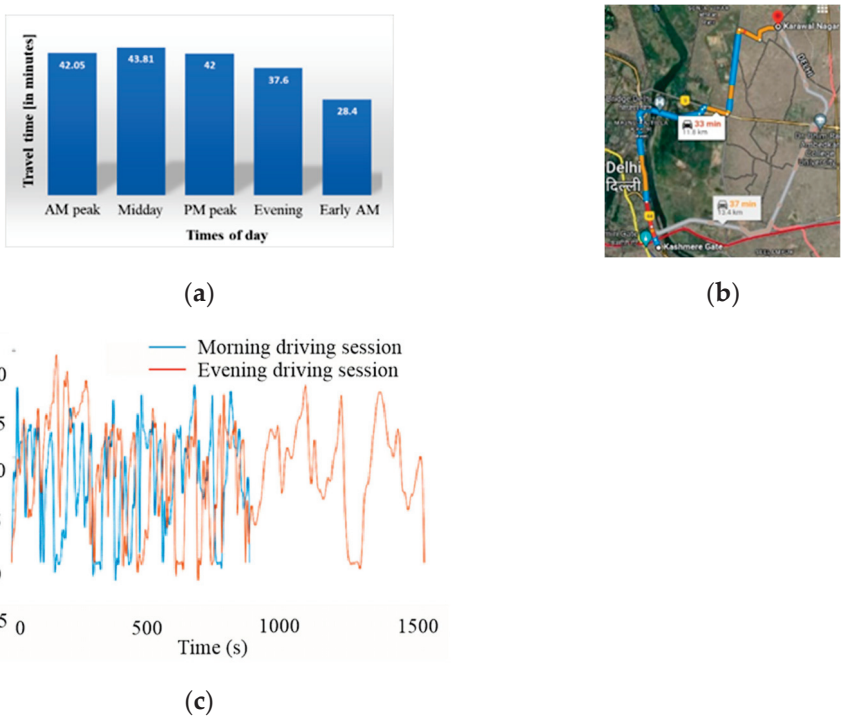


Figure 2. Speed data collection along a route: (**a**) variations in average travel time, (**b**) route map, and (**c**) variations in driving profile.

The collected speed versus time data have been used to extract certain critical characteristic features, such as average speed within a window of 100 s (s^{100}), average positive and negative acceleration within a window of 10 s ($a^{10}+$ and $a^{10}-$), and 5 s ($a^{5}+$ and $a^{5}-$), respectively, along with attributes like number of stops, time of the day, and day of the week. For the classification of roadway type, a sampling time of 100 s is chosen to calculate the average speed. For the classification of traffic type, within the 100 s sampling interval, the data are further sub-sampled for intervals of 10 and 5 s.

In the next step, target classes are defined and IF-ELSE rules are formulated for the classification of the target classes, as shown in Figure 3.

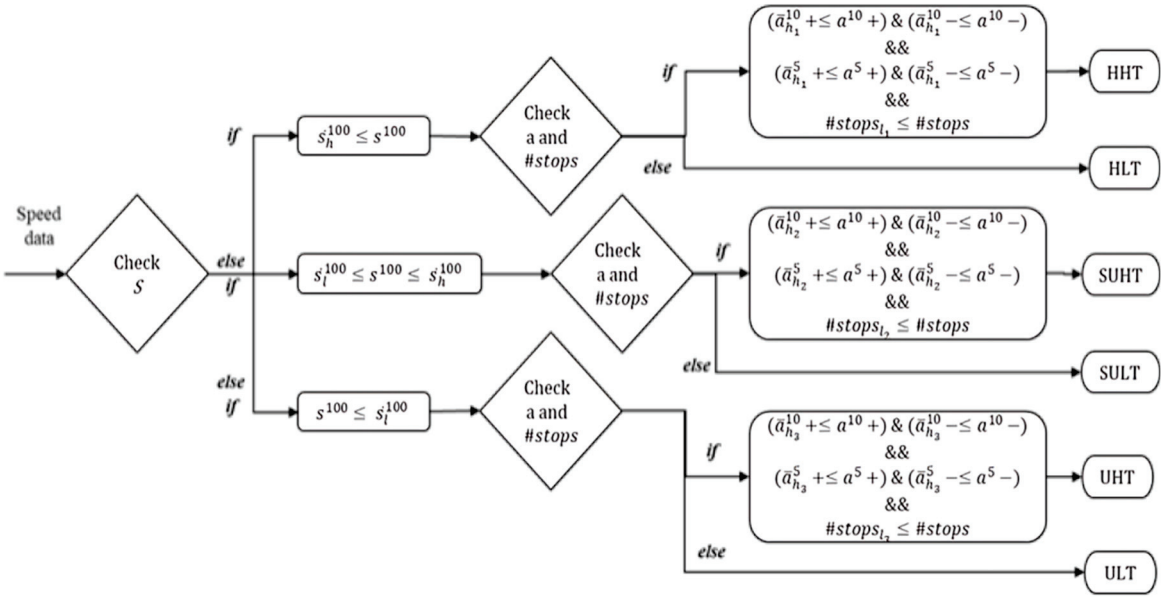


Figure 3. Flow diagram describing different target classes.

The six target classes considered are target class 1: highway high traffic (HHT); target class 2: highway low traffic (HLT); target class 3: suburban high traffic (SUHT); target class 4: suburban low traffic (SULT); target class 5: urban high traffic (UHT); and target class 6: urban low traffic (ULT). When sampled at 100 s, the first condition checked is that of average speed. The comparison of a 100 s average speed with a higher (s_h^{100}) and lower bound (s_l^{100}) gives an estimate about the roadway type. The next conditions which are checked are related to the traffic condition, namely, $a^{10}+$, $a^{10}-$, a^5+, a^5- and number of stops (#stops).

Case 1: If $s_h^{100} \leq s^{100}$, then an estimate is made about the roadway type being a highway. If $a_h^{10}+ \leq a^{10}+$, $a_h^{10}- \leq a^{10}-$, $a_h^5+ \leq a^5+$, $a_h^5- \leq a^5-$, and $\#stops_l \leq$ #stops, the traffic condition is considered to be high traffic, and otherwise low traffic type.
Case 2: If $s_l^{100} \leq s^{100} \leq s_h^{100}$ then the road condition is considered to be suburban. If $a_h^{10}+ \leq a^{10}+$, $a_h^{10}- \leq a^{10}-$, $a_h^5+ \leq a^5+$, $a_h^5- \leq a^5-$, and $\#stops_l \leq$ #stops, then the traffic condition is considered to be high traffic, and otherwise low traffic type.
Case 3: If $s^{100} \leq s_l^{100}$ then the road condition is considered to be urban. If $a_h^{10}+ \leq a^{10}+$, $a_h^{10}- \leq a^{10}-$, $a_h^5+ \leq a^5+$, $a_h^5- \leq a^5-$, and $\#stops_l \leq$ #stops, then the traffic condition is considered to be high traffic, and otherwise low traffic type.

In each case, the upper and lower thresholds are defined separately by taking into consideration the values associated with the standard cycles. Further, an analysis of four standard driving cycles was carried out in terms of the characteristic features, and the results are listed in Table 1. The table reveals the characteristic features of New York city cycle (NYCC), Artemis urban, urban dynamometer driving schedule (UDDS), and highway fuel economy test cycle (HWFET).

Based on the data collected under real driving conditions and the standard cycle data, synthetic datasets were generated and utilized to create RDPs. RDPs represent different driving conditions for roadway and traffic type and have been developed based on comparisons of characteristic features of real-time driving and standard cycles. On the basis of the characteristic features, four standard cycles were compared with the real data points to synthesize the pattern of RDPs. The six RDPs are shown in Figure 4.

Table 1. Characteristic features of standard drive cycles.

Characteristic Features	Different Standard Drive Cycles			
	NYCC	**Artemis Urban**	**UDDS**	**HWFET**
Avg. Drive Speed (m/s)	4.62	6.19	10.16	21.6
No. of Stops	7	14	14	1
S.D. (σ) of Speed (m/s)	3.39	4.46	5.96	4.44
5 s Avg. Speed (m/s)	3.19	4.91	8.75	21.577
10 s Avg. Speed (m/s)	3.21	4919	8.79	21.71
100 s Avg. Speed (m/s)	3.11	4.7	8.98	21.78
1 s Avg. Acceleration (m/s^2)	3.19	4.91	8.75	21.577
5 s Avg. Acceleration (m/s^2)	3.21	4.919	8.79	21.71
10 s Avg. Acceleration (m/s^2)	3.11	4.77	8.98	21.78

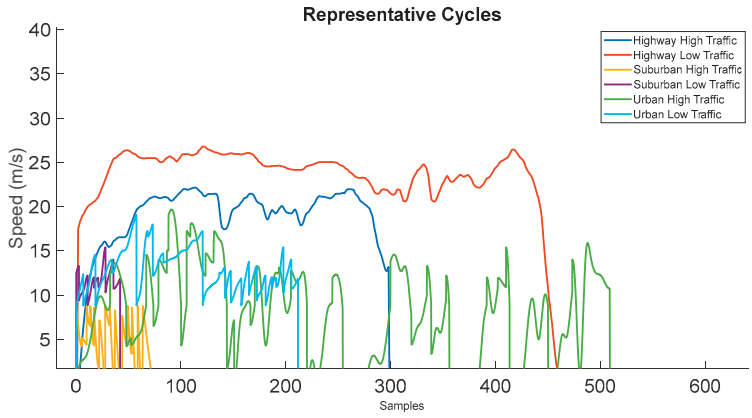


Figure 4. RDPs for different road and traffic conditions.

2.2. Classification Employing MLPNN

The main task of classification is to identify an unknown driving cycle and translate it into one of the known target classes. Classification was performed using MLPNN, with specifications as given in Table 2.

Table 2. Hyperparameters of MLPNN classifier.

Hyperparameters	Specifications
Input layer	10 inputs
Hidden layer	4 neurons
Output layer	6 outputs
Activation function	ReLU
Loss function	Cross-entropy
Learning rate	Function of quadratic approximation of loss
Optimizer	Gradient descent
Training data	19,922 speed data values
Train-test split	80–20%

The complete process of DPR is shown in Figure 5. After the data acquisition and pre-processing steps, a driving cycle database was used to train the MLPNN classifier. A

classification into the six target classes was performed with an accuracy of 97.7% [23]. The RDP was used to select a pattern corresponding to the classified target classes. The selected pattern was then used to compute and optimize the energy consumption of the EV.

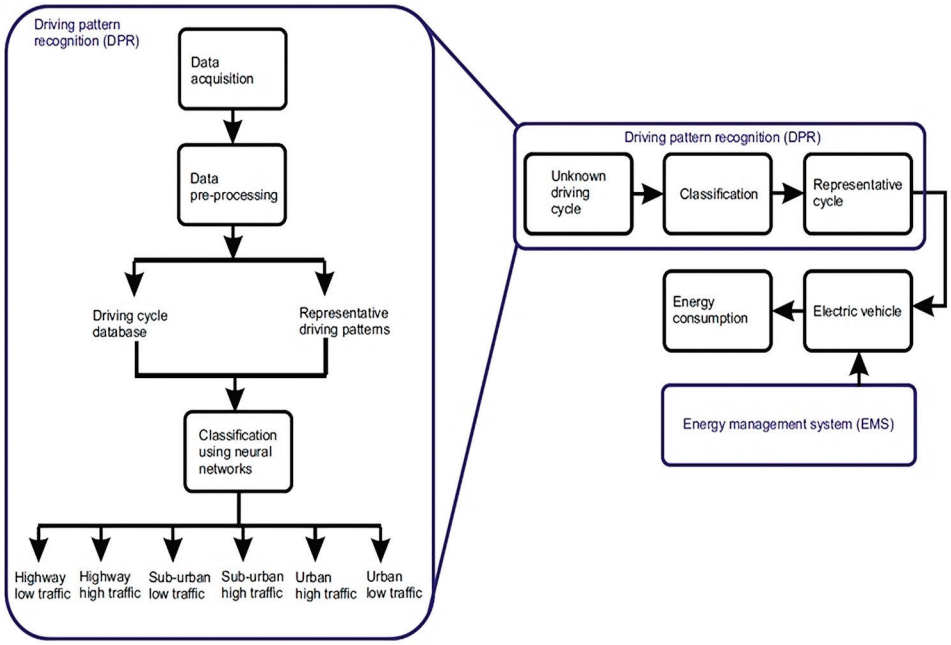


Figure 5. Integration of EMS with DPR.

2.3. Energy Consumption of the EV

An EV model has been implemented in MATLAB-Simulink 2018a to analyze the energy consumption under different identified road and traffic conditions. An EV model consisting of vehicle dynamics, battery, and electric drive is considered with specifications in Table 3.

Table 3. EV model specifications.

Parameter	Specifications
Vehicle mass	2000 kg
Frontal area	1.6 m^2
Aerodynamic drag coefficient	0.3
Rolling resistance coefficient	0.01
Wheel radius	0.28 m
Battery capacity	70 kWh

The total tractive force is given as [24]

$$F = ma + k_1 v^2 + k_2 mg + mgsin\theta \quad (1)$$

where the mass is represented by m and acceleration by a. The velocity is given by v and k_1 and k_2 are constants. The acceleration due to gravity is given by g. The tractive power is given as the sum of electrical losses and mechanical forces as

$$P = I^2 r + Fv. \quad (2)$$

where I represents the motor current demand and r the motor resistance. The total power (P_{total}) is the sum of the electrical power loss, mechanical resistive losses and possible energy gained/lost from regeneration/driving, as

$$P_{total}(v,a) = \frac{rR^2}{K^2}\Big(ma + k_1v^2 + k_2mg + mgsin\theta\Big) + v\Big(k_1v^2 + k_2mg + mgsin\theta\Big) + mav, \quad (3)$$

where K is given by $K_a\Phi_d$, with K_a representing the armature constant and Φ_d the magnetic flux. The radius of the tire is represented by R.

The energy consumption may be calculated as

$$E_{total} = \int P_{total} \cdot dt. \quad (4)$$

2.4. EMS Problem Formulation Using SQP

Determination of optimal speed trajectory for minimum energy consumption by the vehicle is one of the targets of eco-driving. An SQP approach has been considered in this regard [25–27], as it scores over the dynamic optimization approach with a requirement of a lower computational burden. It is also better than the static optimization approach, as it can incorporate non-linearities. The eco-driving problem is formulated as

$$f(N) = min\int_{t_0}^{t_f} P_{total}(v(t), a(t))t. \quad (5)$$

where f represents the objective function, with N as the target class representing HHT, HLT, SUHT, SULT, UHT, and ULT. The upper and lower bounds are as

$$\underline{v}(t) \leq v(t) \leq \overline{v}(t)$$

$$\underline{a}(t) \leq a(t) \leq \overline{a}(t).$$

Equations (1)–(4) can be converted to a finite-dimensional optimization problem by using Euler's discretization, as

$$\min x_k, u_k \sum_{k\in K} \frac{1}{2}\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H_k \begin{bmatrix} x_k \\ x_k \end{bmatrix} + F_k^T \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \quad (6)$$

$$x_{k+1} = f(x_k, u_k),$$

$$\underline{x}_k \leq x_k \leq \overline{x}_k,$$

$$\underline{u}_k \leq u_k \leq \overline{u}_k.$$

where $k \in K = \{0, 1, 2, \ldots K-1\}$, with K being the optimization horizon, $u = \{u_0, \ldots u_k - 1\}$ being the input variables or decision variables, and $x = \{x_0, \ldots x_k\}$ being the state variables.

Here,

$$H_k = \tau\begin{bmatrix} \gamma_0 & 0 & \gamma_1 \\ 0 & 0 & 0 \\ \gamma_1 & 0 & \gamma_2 \end{bmatrix}, F_k = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, x_k = \begin{bmatrix} v_k \\ a_k \end{bmatrix} \quad (7)$$

where τ is the step size and $\gamma_0, \gamma_1, \gamma_2$ are non-negative model parameters.

As H_k is not positive definite, the problem is non-convex. In order to reach a global minimum, SQP aims to solve the non-convex or non-linear problem by sequentially solving linearly constrained quadratic programs of (6), as

$$\begin{aligned} &\left\{x_k^{i+1} u_k^{i+1}\right\}_{k\in K} = argmin x_k, u_k \\ &\sum_{k\in K} \frac{1}{2} \begin{bmatrix} x_k - x_k^i \\ u_k - u_k^i \end{bmatrix}^T R_k \begin{bmatrix} x_k - x_k^i \\ u_k - u_k^i \end{bmatrix} + \left(H_k \begin{bmatrix} x_k^i \\ u_k^i \end{bmatrix} + F_k \right)^T \begin{bmatrix} x_k \\ u_k \end{bmatrix} \end{aligned} \tag{8}$$

Subject to linear states,

$$x_{k+1} = f\left(x_k^i, u_k^i\right) + \nabla f\left(x_k^i, u_k^i\right) \begin{bmatrix} x_k - x_k^i \\ u_k - u_k^i \end{bmatrix} \tag{9}$$

such that the SQP subproblems (8) and (9) are convex and convergent. Here, R_k is added to ensure strict convex behavior.

3. Results and Discussion

In this section, the proposed DPR-based EMS is evaluated in different stages. First, a standard cycle comprising mixed individual known driving cycles is classified with the DPR and the formation of the RDPs is verified. Next, test data collected from two different routes at two different timings are used to evaluate the accuracy of the model. The effect of increasing time frame on classification accuracy is also evaluated. As a next step, a sensitivity analysis is conducted by varying the thresholds of speed and acceleration. Moreover, the relative importance of features on the classification result is analyzed. Finally, the proposed DPR with the EMS is implemented under varying driving conditions.

The simulation was performed on an Intel (R) Core (TM) system with i5-1155G7 at 2.50 GHz. The system had a 64-bit operating system, with a x64-based processor. For real-time implementation, two options of on-board (Edge AI) or off-board processing (Cloud AI) could be considered. Embedded GPUs (Graphics Processing Units) and higher RAM capacities are required for advanced deep learning models with 10–15 trillion operations per second. Cloud AI is useful in offloading heavy computations from the vehicle and is particularly useful in the optimization of high-performing neural networks.

A mixed driving cycle is developed by combining individual standard driving cycles to verify the accuracy of RDPs and DPR processes. A training accuracy of 97.7% is achieved with the classification of unknown speed profiles into the known RDPs. In Figure 6, it can be observed that the mixed cycle comprising sections from standard cycles can be successfully classified into the respective target classes.

Next, driving data collected over two test routes during early morning and evening, with specifications given in Table 4, are analyzed. The speed and acceleration over time show the variations in pattern due to road and traffic conditions, as shown in Figure 7. The positive and negative values indicate positive and negative acceleration, respectively. The accuracy of classification for both routes at the two timings are shown in Figure 8. The validation accuracy is slightly less than the training accuracy. The accuracy is higher for test route 1 than test route 2, mainly due to the fact that the speed variations are higher in test route 2. Moreover, with heavier traffic in the evening, the accuracy values are less than those in the morning. This can be solved by considering different time horizons or sampling intervals.

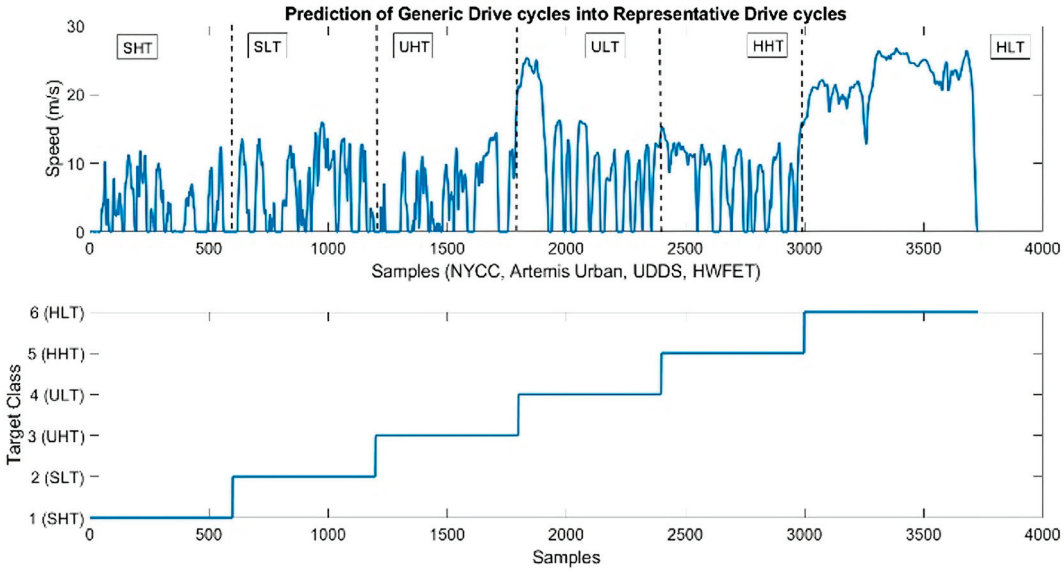


Figure 6. Classification of mixed cycle into RDPs.

Table 4. Specifications of data collection over test routes.

Test Routes	Source	Destination	Distance Traveled	Road Type	Time of Data Collection
Test route 1	12.58532° N 77.4335° E	12.5523° N 77.4106° E	12.4 km	Urban	Early morning and evening
Test route 2	23.0306° N 88.1351° E	23.0962° N 88.0814° E	15.2 km	Highway	Early morning and evening

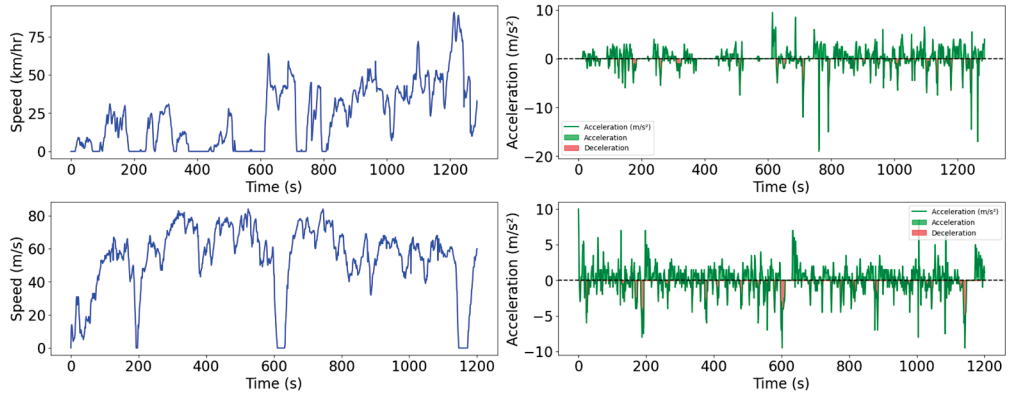


Figure 7. Driving speed and acceleration during (**a**) test route 1 and (**b**) test route 2.

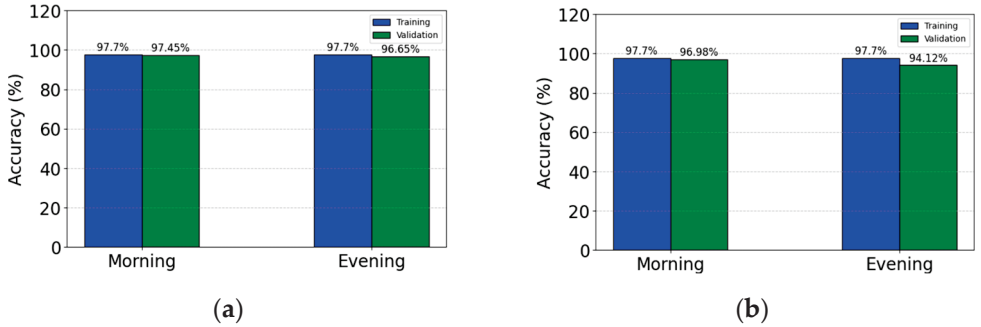


Figure 8. Accuracy in % for morning and evening data for (**a**) test route 1 and (**b**) test route 2.

The timing of data collection is essential for understanding the influence of traffic. The labels of test route 1 during evening and early morning hours are shown in Figure 9. As noted from the figure, the data during the evening have a higher percentage of high traffic regions and hence more times of occurrence of UHT. The data during the morning have more frequent occurrence of ULT.

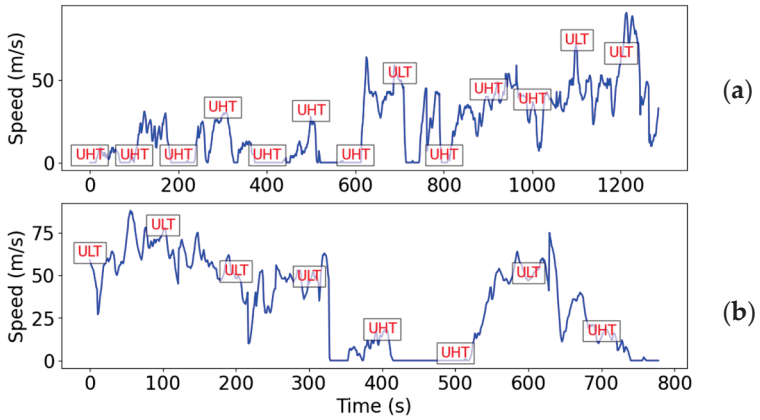


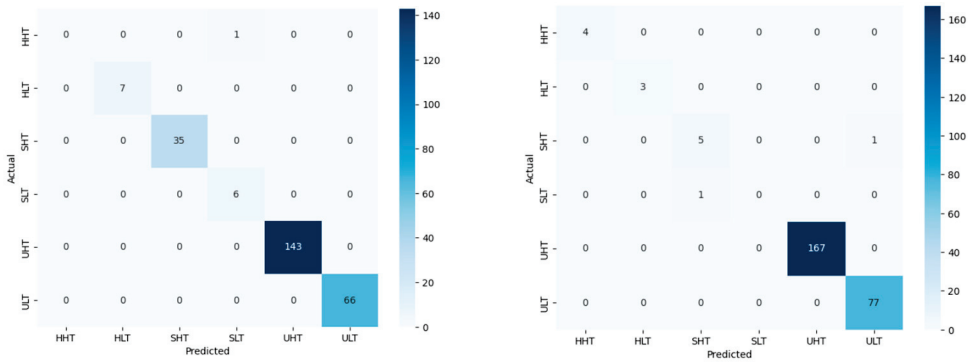
Figure 9. Labels for test route 1 sampled every 100 s for (**a**) evening and (**b**) early morning.

The sampling interval for acceleration data changed from 100 s to 10 s and then to 5 s. The corresponding values of training and validation accuracy are recorded in Table 5. An effect of varying the time horizon of the acceleration data can be observed. For the data from early morning, decreasing the time horizon from 100 s to 10 s led to significantly higher accuracy, but decreasing it from 10 s to 5 s did not bring much improvement. This is due to the fact that although analyzing shorter segments is useful in predicting traffic conditions, too much shortening does not lead to meaningful results. For the data from the evening, the improvement in accuracy due to a decrease in time horizon is more significant than in the case of the morning dataset. Decreasing the time horizon to 5 s, however, brings a slight improvement.

Table 5. Accuracy and loss over 100 s speed samples for test route 1 with varying acceleration samples.

Speed Sampling Interval	Acceleration Sampling Interval	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Data collection timing: early morning					
100 s	100 s	0.7750	0.9920	0.7566	0.8190
	10 s	0.9777	0.0229	0.9622	0.0239
	5 s	0.9745	0.0265	0.9961	0.0260
Data collection timing: evening					
100 s	100 s	0.4580	0.8997	0.5632	0.8921
	10 s	0.9777	0.0452	0.9665	0.1479
	5 s	0.9710	0.0322	0.9691	0.1098

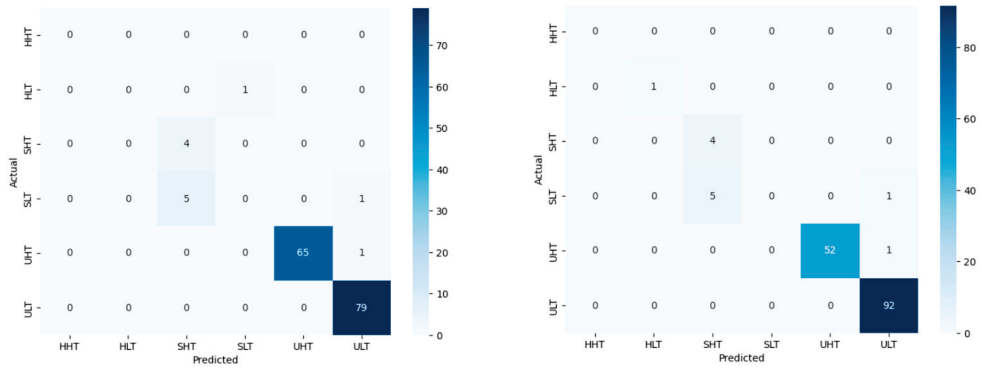
As a next step, the influence of speed and acceleration thresholds on the classification accuracy were analyzed. In Figure 10, the confusion matrix for test route 1 during the evening session is shown. Increasing the threshold values led to more data being correctly classified. When threshold values were increased by 5%, 167 samples were classified as UHT and 77 as ULT, as shown in the confusion matrix. In this case, five samples were wrongly classified as SHT, three as HLT, and four as HHT. When threshold values decreased by 5%, 143 samples were classified as UHT and 66 as ULT. In this case, 35, 6 and 7 samples were wrongly classified as SHT, SLT and HLT, respectively.



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

Figure 10. Confusion matrix for test route 1 during the evening with threshold values: (**a**) decreased by 5%; (**b**) increased by 5%.

In Figure 11, the confusion matrix for test route 1 during the morning session is shown. Increasing the threshold values led to more data being correctly classified. When threshold values increased by 5%, 92 samples were classified as ULT and 52 as UHT. In this case, four samples were wrongly classified as SHT and one as HLT. When threshold values decreased by 5%, 79 and 65 samples were classified as ULT and UHT, respectively. In this case, four samples were wrongly classified as SHT.



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

Figure 11. Confusion matrix for test route 1 during the morning with threshold values: (**a**) decreased by 5%; (**b**) increased by 5%.

The relative importance of the four features, namely speed, positive and negative acceleration, and number of stops, was evaluated through F-score and p-value to measure the effectiveness with which each of these features separates the different classes. As shown in Table 6, for both routes, speed is the most determining feature for classification as its F score value is the highest. For test route 1, positive and negative acceleration do not play such vital roles as in test route 2. This is because this route has higher changes in acceleration and deceleration values. Moreover, the number of stops has a higher occurrence in test route 1 than test route 2, and hence a higher F score value in test route 1. The *p*-value indicates the relevancy of the feature for classification. Since p-values are less than 0.05 for all features, they are all relevant in the classification process.

Table 6. F-score and *p*-value for the different features over the two test routes.

Test route 1		
	F-score	*p*-value
Speed	612.2230	0.00145
Positive acceleration	123.9876	0.00169
Negative acceleration	97.8761	0.00213
No. of stops	12.67	0.0315
Test route 2		
	F-score	*p*-value
Speed	997.2367	0.00345
Positive acceleration	432.2166	0.00193
Negative acceleration	398.1298	0.00321
No. of stops	2.11	0.0398

The energy management problem solved using SQP provides the reference speed values for optimized energy consumption of the EV. As shown in Figure 12, with the mixed speed profile, the classification using DPR provides information about the type of driving and traffic situation. EMS provides reference speed values corresponding to minimum energy consumption.

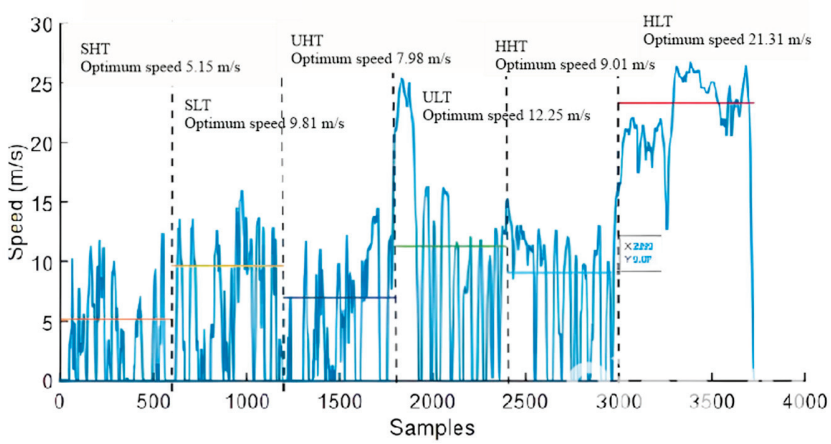


Figure 12. Reference speed and energy values.

The energy consumption of the vehicle and the battery state of charge (SOC) are shown in Figures 13 and 14, respectively. The energy is observed to be increasing over time and the battery SOC has been depleted to a minimum value over time.

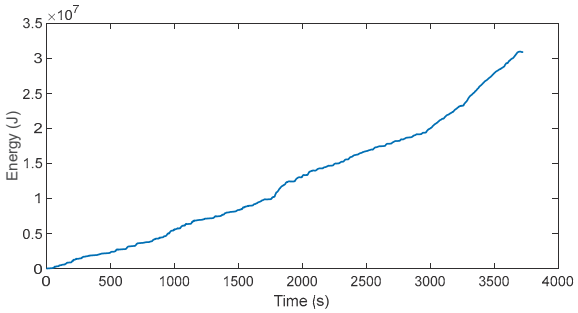


Figure 13. Energy consumption with the mixed cycle.

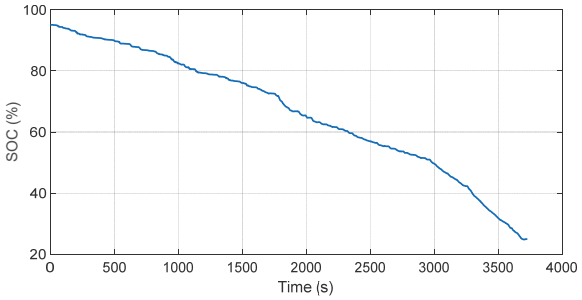


Figure 14. Battery SOC with the mixed cycle.

Table 7 depicts the energy consumption of the six RDPs obtained through analysis of the EV model. For the highway type of roadway, the energy consumption falls under the range of 3.825–7.96 kWh for different road traffic conditions. Similarly, for the suburban and urban types of roadways, the energy consumption lies in the range of 0.985–0.964 kWh and 2.87–5.46 kWh, respectively. It can be observed that the error in the case of energy consumption obtained under real driving conditions and as calculated by the RDPs is relatively small.

Table 7. Energy consumption and error calculation.

RDPs	Energy Consumption (kWh)	Error Observed (kWh)
Highway High Traffic	3.825	5.1
Highway Low Traffic	7.96	7.2
Suburban High Traffic	0.985	0.9
Suburban Low Traffic	0.964	0.5
Urban High Traffic	5.46	3.3
Urban Low Traffic	2.87	2.7

Figure 15 depicts a comparison between the energy consumption predicted with and without applying DPR.

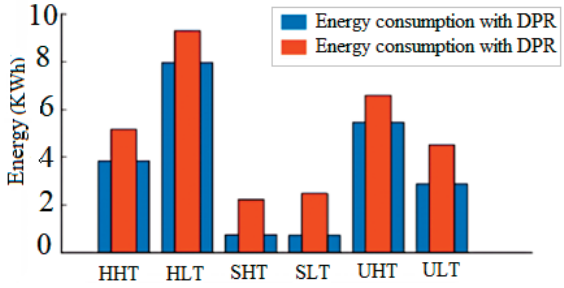


Figure 15. Energy consumption as predicted by the RDPs.

Table 8 further elaborates the optimum speed and average value of optimum energy consumption obtained with EMS from Tables 7 and 8. It is observed that the total energy consumption without EMS is significantly higher than optimum energy (with EMS).

Table 8. Optimal speed and energy consumption.

Representative Pattern	Optimum Speed (m/s)	Optimum Energy (kWh)
Highway High Traffic	9.07	3.1356
Highway Low Traffic	23.31	4.94
Suburban High Traffic	5.171	0.565
Suburban Low Traffic	9.61	0.552
Urban High Traffic	6.996	2.94
Urban Low Traffic	11.25	1.351

Integration of DPR with EMS may lead to a significant improvement in energy savings. However, several practical challenges prevent the successful implementation of this technology. Firstly, large datasets are required, considering multiple factors related to the vehicle and the environment under which the vehicle is being driven, and how the vehicle is being driven. Secondly, low-latency and lightweight solutions are required to ensure the real-time applicability of the system. Thirdly, generalizability over multiple regions with varying conditions requires an adaptive approach. Finally, dynamic unknown factors related to the traffic, accidents, roadblocks, and driving habits need to be considered.

4. Conclusions

In this contribution, a two-layer DPR has been developed to classify an unknown driving cycle into one of the predefined templates or RDPs. The DPR has been implemented with an EV model and SQP-based optimized EMS. The predicted energy consumption has been analyzed along with the implementation of EMS to achieve an optimal speed profile for minimizing energy consumption. The simulation results indicate that the DPR is able to generalize well over different routes and traffic conditions. However, training with a larger database with data from multiple regions under varied traffic conditions can improve the testing accuracy further. A consideration of other factors, such as weather, driving style, unknown incidents during the drive, and charging patterns of the EV, may also provide more insight into the applicability of the DPR. As concluded from the present work, DPR-based EMS can provide lower energy consumption. The optimal speed profile can either be maintained by virtue of eco-driving or provided as advice to the human driver. As part of future work, complete control loop may be implemented to realize eco-driving. More advanced artificial intelligence (AI) tools may be implemented to improve the accuracy of classification. A hardware application can also be considered with on-board (edge-AI) and off-board (cloud AI) systems.

Author Contributions: Conceptualization, B.M.; methodology, B.M.; validation, B.M. and S.K.; formal analysis, M.I.; resources, B.M. and S.K.; writing—original draft preparation, B.M. and S.K.; writing—review and editing, M.I.; supervision, M.I.; project administration, M.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data can be made available upon reasonable request from the authors.

Conflicts of Interest: The authors declare that they do not have any conflicts of interest, and nor do they have any financial or non-financial interests that are directly or indirectly related to the work submitted for publication.

References

1. Al-Dhaifallah, M.; Ali, Z.M.; Alanazi, M.; Dadfar, S.; Fazaeli, M.H. An efficient short-term energy management system for a microgrid with renewable power generation and electric vehicles. *Neural Comput. Appl.* **2021**, *33*, 16095–16111. [CrossRef]
2. Aldosary, A.; Rawa, M.; Ali, Z.M.; Latifi, M.; Razmjoo, A.; Rezvani, A. Energy management strategy based on short-term resource scheduling of a renewable energy-based microgrid in the presence of electric vehicles using θ-modified krill herd algorithm. *Neural Comput. Appl.* **2021**, *33*, 10005–10020. [CrossRef]
3. Zhang, F.; Xiao, L.; Coskun, S.; Pang, H.; Xie, S.; Liu, K.; Cui, Y. Comparative study of energy management in parallel hybrid electric vehicles considering battery ageing. *Energy* **2023**, *264*, 123219. [CrossRef]
4. Kong, H.; Yan, J.; Wang, H.; Fan, L. Energy management strategy for electric vehicles based on deep Q-learning using Bayesian optimization. *Neural Comput. Appl.* **2020**, *32*, 14431–14445. [CrossRef]
5. Son, N.N.; Van Cuong, N. Neuro-evolutionary for time series forecasting and its application in hourly energy consumption prediction. *Neural Comput. Appl.* **2023**, *35*, 21697–21707. [CrossRef]
6. Zhang, J.; Xu, F.; Zhang, Y.; Shen, T. ELM-based driver torque demand prediction and real-time optimal energy management strategy for HEVs. *Neural Comput. Appl.* **2020**, *32*, 14411–14429. [CrossRef]
7. Ali, M.A.; Moulik, B. On the role of intelligent power management strategies for electrified vehicles: A review of predictive and cognitive methods. *IEEE Trans. Transp. Electrif.* **2022**, *1*, 368–383. [CrossRef]
8. Hu, J.; Niu, X.; Jiang, X.; Zu, G. Energy management strategy based on driving pattern recognition for a dual-motor battery electric vehicle. *Int. J. Energy Res.* **2019**, *43*, 3346–3364. [CrossRef]
9. Zhou, B.; Zhao, Z. Multi-objective optimization of electric vehicle routing problem with battery swap and mixed time windows. *Neural Comput. Appl.* **2022**, *34*, 7325–7348.
10. Zhang, S.; Xiong, R. Adaptive energy management of a plug-in hybrid electric vehicle based on driving pattern recognition and dynamic programming. *Appl. Energy* **2015**, *155*, 68–78. [CrossRef]
11. Hu, J.; Liu, D. Intelligent energy management strategy of hybrid energy storage system for electric vehicle based on driving pattern recognition. *Energy* **2020**, *198*, 117298. [CrossRef]
12. Zhang, R.; Tao, J.; Zhou, H. Fuzzy optimal energy management for fuel cell and supercapacitor systems using neural network based driving pattern recognition. *IEEE Trans. Fuzzy Syst.* **2019**, *27*, 45–57. [CrossRef]
13. Wei, Z.; Xu, Z.; Halim, D. Study of HEV power management control strategy based on driving pattern recognition. *Energy Procedia* **2016**, *88*, 847–853. [CrossRef]
14. Kandi Dayeni, M.; Soleymani, M. Intelligent energy management of a fuel cell vehicle based on traffic condition recognition. *Clean Technol. Environ. Policy* **2016**, *18*, 1945–1960. [CrossRef]
15. Zhou, Y.; Ravey, A.; Péra, M.C. Multi-mode predictive energy management for fuel cell hybrid electric vehicles using Markov driving pattern recognizer. *Appl. Energy* **2020**, *258*, 114057. [CrossRef]
16. Lei, Z.; Cheng, D.; Liu, Y.; Qin, D.; Zhang, Y.; Xie, Q. A dynamic control strategy for hybrid electric vehicles based on parameter optimization for multiple driving cycles and driving pattern recognition. *Energies* **2017**, *10*, 54. [CrossRef]
17. Keulen, T.V.; De Jager, B.; Kessels, J.; Steinbuch, M. Energy management in hybrid electric vehicles: Benefit of prediction. *IFAC Proc.* **2010**, *43*, 264–269. [CrossRef]
18. Yokoi, Y.; Ichikawa, S.; Doki, S.; Okuma, S.; Naitou, T.; Shiimado, T.; Miki, N. Driving pattern prediction for an energy management system of hybrid electric vehicles in a specific driving course. In Proceedings of the 30th Annual Conference of IEEE Industrial Electronics Society, Busan, Republic of Korea, 2–6 November 2004; Volume 2, pp. 1727–1732.
19. He, H.; Sun, C.; Zhang, X. A method for identification of driving patterns in hybrid electric vehicles based on a LVQ Neural Network. *Energies* **2022**, *9*, 3363–3380. [CrossRef]
20. Kajiwara, S. Driver-condition detection using a thermal imaging camera and neural networks. *Int. J Automot. Technol.* **2021**, *22*, 1505–1515. [CrossRef]
21. Mehranfar, S.; Banagar, I.; Moradi, J.; Andwari, A.M.; Könnö, J.; Gharehghani, A.; Rabiei, M.; Kurvinen, E. The perspective of energy storage systems advancements and challenges for electric vehicle applications; metric, mechanism, mode, and mitigation framework. *Future Sustain.* **2024**, *2*, 22–32. [CrossRef]
22. Habib, A.R.R.; Butler, K. Environmental and economic comparison of hydrogen fuel cell and battery electric vehicles. *Future Technol.* **2022**, *1*, 25–33. [CrossRef]
23. Moulik, B.; Kaur, S.; Ranjan, S.; Dixit, S. Analysing the Energy Consumption of Urban Electric Vehicles for Real Driving Scenarios. *IETE J. Res.* **2024**, *70*, 5799–5809. [CrossRef]
24. Li, M.; Wu, X.; He, X.; Yu, G.; Wang, Y. An eco-driving system for electric vehicles with signal control under V2X environment. *Trans. Res. Part C Emerg. Technol.* **2018**, *93*, 335–350.
25. Zhang, Y.; Liu, Y.; Huang, Y.; Chen, Z.; Li, G.; Hao, W.; Cunningham, G.; Early, J. An optimal control strategy design for plug-in hybrid electric vehicles based on internet of vehicles. *Energy* **2021**, *228*, 120631.

26. López-Rojas, A.D.; Cruz-Villar, C.A. Neural networks as an approximator for a family of optimization algorithm solutions for online applications. *Neural Comput. Appl.* **2024**, *36*, 3125–3140.
27. Khalik, Z.; Padilla, G.P.; Romijn, T.C.J.; Donkers, M.C.F. Vehicle energy management with ecodriving: A sequential quadratic programming approach with dual decomposition. In Proceedings of the 2018 Annual American Control Conference (ACC), Milwaukee, WI, USA, 27–29 June 2018; pp. 4002–4007.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Protecting Intellectual Security Through Hate Speech Detection Using an Artificial Intelligence Approach

Sadeem Alrasheed [1], Suliman Aladhadh [1,*] and Abdulatif Alabdulatif [2]

[1] Department of Information Technology, College of Computer, Qassim University, Buraydah 52571, Saudi Arabia; 431214114@qu.edu.sa
[2] Department of Computer Science, College of Computer, Qassim University, Buraydah 52571, Saudi Arabia; ab.alabdulatif@qu.edu.sa
* Correspondence: s.aladhadh@qu.edu.sa

Abstract

Online social networks (OSNs) have become an integral part of daily life, with platforms such as X (formerly Twitter) being among the most popular in the Middle East. However, X faces the problem of widespread hate speech aimed at spreading hostility between communities, especially among Arabic-speaking users. This problem is exacerbated by the lack of effective tools for processing Arabic content and the complexity of the Arabic language, including its diverse grammar and dialects. This study developed a two-layer framework to detect and classify Arabic hate speech using machine learning and deep learning with various features and word embedding techniques. A large dataset of Arabic tweets was collected using the X API. The first layer of the framework focused on detecting hate speech, while the second layer classified it into religious, social, or political hate speech. Convolutional neural networks (CNN) outperformed other models, achieving an accuracy of 92% in hate speech detection and 93% in classification. These results highlight the framework's effectiveness in addressing Arabic language complexities and improving content monitoring tools, thereby contributing to intellectual security and fostering a safer digital space.

Keywords: hate speech detection; hate speech classification; two-layer approach; machine learning; deep learning

1. Introduction

Currently, online social networks (OSNs) play an important role in people's daily lives and are changing the way they socialize and network [1]. Through OSNs, any type of content can be shared with a large audience within seconds [2]. In this sense, X (formerly Twitter) is one of the most popular OSNs used by Arabic speakers in the Arab world [3], and it has experienced rapid growth since its inception in 2006. For consistency, the term 'tweets" is used in this study to refer to users' posts on X, as this term is widely recognized and continues to be used colloquially. X allows people to interact and maintain connections with their contacts and talk about their daily activities, making it a useful resource and a rich source of information [4].

In addition, the real-time nature of X enables the rapid dissemination of news, trends and opinions that influence public debate and social trends [5]. This makes the platform not only a tool for personal communication, but also an important medium for political, social and cultural exchange [6]. Nevertheless, the rapid spread of the platform and

the nature of social interaction have led to the proliferation of extremist activities that directly or indirectly promote violence, intolerance, racism or other undesirable content. Extremist activities affect individuals and civilizations by influencing ideas and beliefs and inciting hatred and violence. Intellectual cyberthreat patterns emerge through various intellectual vessels and sources that spread deviant and decadent ideas through digital platforms, incite hatred and erase dissenting opinions under the guise of communication, reconciliation and cultural and intellectual integration. Toxic online content can not only cause psychological harm but also radicalize people and lead to actual hate crimes [7]. Hate speech is a form of extremism that is rooted in prejudice and intolerance and often leads to violence, discrimination and persecution. Alhazmi [8] highlights key challenges in identifying hate speech on Arabic Twitter, including the linguistic richness and ambiguity of Arabic and the dearth of comprehensive annotated corpora. The main goal of this study is to open an integrated approach using artificial and deep learning to identify and detect and classify hate speech in Arabic social media, contributing to intellectual security and safer digital spaces. The aim of this study is to develop methods to identify and resolve such problems and provide an improved experience in cyberspace. This paper presents a framework based on pre-training specific embeddings in Arabic, such as AraBERT with a CNN designed to work in two layers. All previous studies have focused on the study of English, while this paper deals with the application of these mechanisms to Arabic, as one of the complex languages, including its rich morphology and dialectal differences. It reflects tensions between different groups within and outside society and can spread rapidly via OSNs [9].

In recent years, extremist actions inciting hate have led to severe incidents, such as those in the United States. Some groups, through OSNs, have undergone what is described as a "self-learning process", which influenced them to adopt beliefs that white supremacy necessitates violent behavior. This has culminated in tragic events including shootings at a black church [10]. The Center for Countering Digital Hate notes that the dissemination of hate speech intensifies significantly during major global events. For instance, during the World Cup, the social media platform X failed to eliminate 99% of hateful messages targeting Qatar and its players [11]. In order to protect intellectual security, the detection of hate speech, a form of intellectual extremism, is crucial, and must be performed effectively and promptly.

Arabic is the fourth most popular language on the Internet and is the sixth most popular language on X [12]. The use of X by Arabic speakers has increased significantly over time, as shown in Figure 1 below.

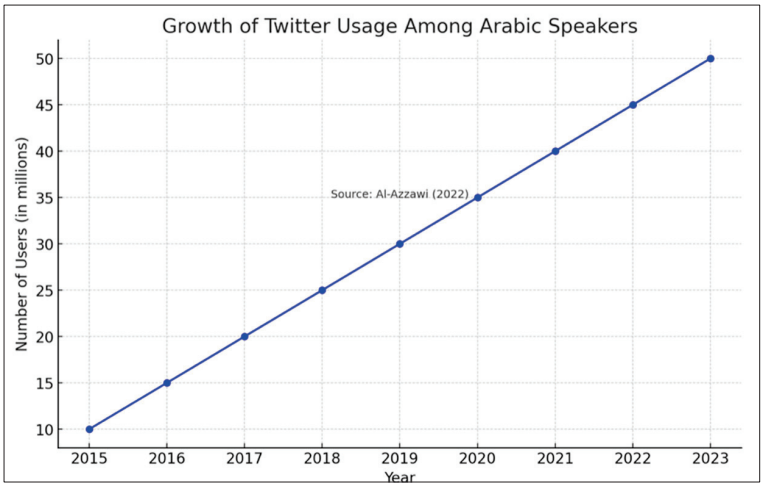


Figure 1. Growth of X usage among Arabic speakers (2015–2023) [12].

This rise highlights the need to address issues such as hate speech within the Arabic-speaking X community. Extensive studies have been conducted on detection of hate speech in different languages, especially English. Nevertheless, few relevant studies have been conducted in the Arabic language due to many limitations, including the complex linguistic features of Arabic, which differ from those of English, and the lack of large datasets. Based on the social responsibility of the individual towards society and the widespread accessibility of social media, this study makes the following important contributions:

- Develops a system to detect hate speech in Arabic and classify it into specific types, with the goal of protecting intellectual security in society.
- Creates a comprehensive Arabic hate speech dataset to support accurate detection and classification of hate speech across different dialects and contexts. This dataset will be a valuable resource for future research and technological advancements aimed at combating hate speech in the Arabic-speaking online community.

The remainder of this paper is structured as follows: Section 2 provides an overview of the fundamental concepts related to hate speech. Section 3 presents related studies and prior research in the field. Section 4 outlines the materials and methods used in this study. Section 5 details the experimental results and discusses them. Section 6 concludes the paper.

2. Fundamental Concepts of Hate Speech

X offers cheap, instant and decentralized distribution. Due to these characteristics, X has become an important medium for hate groups, as it is used to spread propaganda, inform about their targets and exchange information between like-minded people. It also allows them to justify the use of violence and legitimize their actions as they work to obstruct and delegitimize others. With the growing popularity of X, the opportunities to spread hate through this platform are multiplying [13]. Hate speech, as defined in Section 1, aims to harm target groups by insulting, harassing, intimidating, humiliating and victimizing them. It also aims to promote discrimination and hostility towards these groups based on innate or perceived characteristics such as gender, ethnicity, religion or sexual orientation. On the other hand, a hate environment is defined as an environment that publishes a hate message in any form, be it textual, visual or audio [14]. Its main aim is to spread hostile ideas and discriminatory attitudes online, which increases the severity and seriousness of this phenomenon and contributes to fueling hatred and extremism in digital societies.

Types of Hate Speech

As hate speech continues to rise on digital platforms, it is essential to comprehend its nature, in order to identify the types of attacks that different platforms encounter. This understanding aids in developing effective strategies to combat digital hate and in implementing appropriate measures to protect users, ensuring a safe communication environment. We will examine three main types of hate: political hate, social hate, and religious hate. These types of hate have been systematically researched in systematic reviews, including, but not limited to, Jahan and Oussalah [15], where the authors discuss how NLP-based approaches remain critical for such recognition.

Political hate speech (PHS) occurs when people or groups are attacked because of their political orientation or opinion. This usually occurs in times of political crisis or during elections, where hostile or insulting language is used against political opponents. Political hatred can take the form of insults, threats or questioning the loyalty of others to their home country. This type of hatred can sometimes lead to heightened political tensions and cause outbreaks of violence within the community [16].

Social hate speech (SHS) is directed against specific groups within a society, based on social characteristics such as ethnicity, social class, gender identity or even disability. This type of discourse aims to consolidate discrimination and social dominance by deepening the differences between different groups and excluding some individuals or groups from social and economic life. Most of the time, this type of language is used to spread negative thoughts or stereotypes that lead to the division of society and the exacerbation of differences between classes or social groups [17].

Religious hate speech (RHS) is directed against individuals or groups because of their religious beliefs, often in connection with differences in doctrine or cultural differences arising from religion. Religious hatred manifests itself in many forms, including discrimination in treatment, incitement to violence against religious communities, or the dissemination of hate speech against adherents of a particular religion. This type of hatred is considered one of the most dangerous as it affects religious identity, which is an integral part of the identity of individuals and groups [18].

Each of these types has particular characteristics and affects specific groups, but all can lead to the destruction of social cohesion and increase conflict. Understanding the different types of hatred helps to develop effective strategies to combat it and prevent its spread.

3. Related Studies

The detection of hate speech has become an important area of research, due to its impact on social cohesion and online safety. Efforts in this area have focused on developing systems and datasets to detect and classify hate speech in different languages and contexts. Approaches differ significantly between non-Arabic and Arabic languages due to linguistic, cultural and contextual differences. This section provides an overview of previous studies on hate speech detection.

3.1. Hate Speech Detection in Non-Arabic Languages

Several studies have explored hate speech detection in different languages using advanced machine learning and deep learning techniques. Gaddisa [19] focused on Afaan Oromo, leveraging tweets and Facebook posts annotated by linguistic experts and testing models such as CNNs, LSTMs, BiLSTMs, GRUs, and CNN-LSTM hybrids, with BiLSTM achieving the highest F1 score of 91%. This study highlighted the benefits of pre-trained embeddings and extended sampling, suggesting broader applicability across languages. Similarly, reference [20] examined Turkish hate speech detection using print media articles, employing models including the Hierarchical Attention Network (HAN) and BERT. BERT achieved the best accuracy of 90.6% when augmented with Turkish-specific linguistic features, underlining its robustness.

For the English language, Djamila et al. [21] utilized neural networks on the HASOC-2021 dataset, comparing BERT-based models, CNNs and ensembles, with the ensemble classifier achieving an F1 score of 85.1%, despite minor misdetections. Ananya et al. [22] addressed hate speech in Hinglish, using context-based embeddings such as ELMo, FLAIR, and BERT, achieving an F1 score of 87%, demonstrating their effectiveness for code-mixed languages. Flor et al. [23] analyzed Spanish hate speech using datasets from HaterNet and SemEval 2019, testing models such as BERT, XLM, and BETO, with F1 scores of 65.8% and 75.5%, respectively.

These findings emphasized the challenges posed by rare words and mislabeling, underscoring the need for improved datasets and model robustness. These studies collectively showcase the potential of advanced models and tailored approaches in addressing hate speech across diverse languages and contexts.

As with other models, research [24,25] has concluded that AraBERT is created to focus on the Arabic language. It is fine-tuned on massive amounts of Arabic text data, bringing in the ability to understand the finer aspects of the Arabic language, including its morphology and syntax. As has been demonstrated, AraBERT outperforms Arabic NLP tasks [26], such as sentiment analysis, named entity recognition and question answering. As a result, this model is a substantial breakthrough in the field of Arabic NLP, as it overcomes the problem of limited access to high-quality Arabic NLP sources and models.

mBERT [27] is expected to deal with several languages at any given time. The model is a variant of BERT and is initialized with a corpus of 104 languages, thus supporting transfer learning. This enables the model to transfer knowledge from one language to another, which enhances performance from one language to the other, and mBERT is useful in multilingual tasks, such as multilingual translation, cross-lingual opinion mining and multilingual question and answer. This novel model alleviates the lack of connection between NLP studies in one language and another, allowing for scholarly unity and cooperative effort.

DistilBERT is one of the variants of BERT [28], which intends to minimize memory consumption and computational time. It is more compact than the 'classic' BERT, it outperforms sacrificing mere parameter count, and has substantially faster inference times. DistilBERT minimizes redundancy and is trained to match outcomes of a larger, teacher model (such as BERT), using knowledge distillation methods. In its current form, this model provides a feasible approach to incorporating BERT-like models to contexts where resources are limited or to situations where responses must be generated as quickly as possible.

3.2. Hate Speech Detection in Arabic Languages

This section focuses on studies conducted specifically on Arabic hate speech detection, providing a detailed overview of the methods, tools and models used.

Nuha et al. [29] compiled a dataset of 6000 Arabic tweets containing religious hate speech, categorizing each tweet into hate or non-hate categories and indicating specific religions for hate tweets [30]. During preprocessing, diacritical marks, punctuation, emojis, non-Arabic characters and single-letter words were removed and normalized. The dataset was analyzed using machine learning models, including logistic regression (LR), support vector machine (SVM) and gated recurrent units (GRU), with the GRU-based model achieving an F1 score of 77%. Hala et al. [31] presented the L-HSAB dataset [32], comprising 5846 tweets labeled as normal, offensive, or hateful, with naive Bayes (NB) achieving the highest F1 score of 74.4%. Hatem et al. [33] developed the T-HSAB dataset for Tunisian Arabic [34], containing 6075 comments divided into normal (3834), offensive (1127), and hate (1078) categories, with NB outperforming SVM, achieving an F1 score of 83.6%.

Bushr et al. [35] used the OSACT4 dataset [36] to identify offensive speech in Arabic, integrating convolutional neural networks (CNN) and bidirectional gated recurrent units (Bi-GRU) enhanced with attention layers, achieving an F1 score of 85%. Omar et al. [37] collected data from multiple OSNs and evaluated 12 machine learning techniques and two deep learning models, with recurrent neural networks (RNN) achieving an F1 score of 98.7%, despite challenges in reproducibility due to limited transparency. Fatemah [38] focused on preprocessing informal social media texts for offensive language detection using the OSACT4 dataset and an SVM model, achieving 90% accuracy, although this was limited by the absence of advanced features during model training. Abdullah et al. [39] utilized the OSACT4 dataset of 10,000 tweets, with long short-term Memory (LSTM) performing best, with an F1 score of 85%.

Wissam et al. [40] used AraBERT with multitasking and multilabel classification techniques on the OSACT4 dataset, achieving a macro-F1 score of 90.15%, although it faced misclassification issues in non-offensive contexts. Raghad et al. [41] developed a dataset of 9316 annotated tweets, with a CNN model achieving an F1 score of 79%, although the study's results were context-specific. Faris et al. [42] achieved a 91% hit rate using a dataset of 3696 tweets processed with term frequency-inverse document frequency (TF-IDF) and bag of words (BoW) methods, but faced challenges with mislabeled data. Amana et al. [43] combined multiple datasets (ArHS, L-HSAB, and OSACT4) for a total of 23,678 tweets, using BiLSTM-CNN to achieve F1 scores of 79%, 74%, and 73% for binary, ternary, and multi-label classifications, respectively.

Faisal [44] developed a dataset of 4203 comments categorized into seven classes, achieving impressive results with deep recurrent neural networks: 99.73% accuracy for binary classification, 95.38% for ternary classification and 84.14% for multi-class classification, although limited by its small dataset size. Hanane et al. [45] used a dataset of 15,050 YouTube comments [46], with the CNN-LSTM model achieving an F1 score of 83.65%, but noted the dataset's outdated nature. Bassma et al. [47] developed a two-stage system using pre-trained word embeddings and a hybrid classification approach evaluated on the ArCybC dataset [48], with the SVM classifier achieving an F1 score of 87.8%, although it was affected by the outdated dataset [49].

Collectively, these studies underscore the importance of tailored approaches and highlight the challenges in Arabic hate speech detection, particularly concerning dataset diversity, dialect variation and evolving linguistic trends. The techniques discussed are summarized in Table 1, while Table 2 shows the only publicly available Arabic datasets used in hate speech detection studies.

Pretrained transformer models such as the BERT family including SBERT continue to deliver remarkable performance in MNLNLPTD because of their contextual understanding over sequence lengths. Some models in Arabic, such as AraBERT, are fine-tuned for task like, sentiment analysis and question answering, but these need high computational power. Most prior work focuses on structured Arabic datasets or English and have had high accuracy; however, issues have not been addressed with the informal variation between Arabic dialects and hate speech language.

Table 1. Summary of research related to hate speech detection.

Rf.	Year	Techniques	Classifier	Language	Datasets	Preprocessing	Max-score
[29]	2018	Machine Learning/ Deep Learning	LR, SVM, GRU	Arabic	Collection of 6000 tweets	Removing (diacritics, punctuation, non-Arabic characters, emoticons, and stop words). Normalizing letters	GRU = 79% Acc.
[31]	2019	Machine Learning	SVM, NB	Arabic	Collection of 5846 tweets	Removing non-Arabic characters, emojis, URLs,	NB = 88.4% Acc.
[33]	2019	Machine Learning	SVM, NB	Arabic	Collection of 6075 comments	Removing non-Arabic characters, emojis, URLs,	NB = 87.9.4% Acc.
[22]	2019	Deep Learning/Machine Learning	BERT, ELMo, FLAIR, Bi-LSTM	English/Hindi	Collection of 8000 tweets, YouTube, Instagram comments	Removing URLs, hashtags, mentions and punctuations	FLAIR BERTMU + Bi-LSTM = 73% Acc.
[35]	2020	Deep Learning	BiGRU and CNN	Arabic	Collection of 10,000 tweets and YouTube comments	Removing (diacritics, punctuation, non-Arabic characters, emoticons, and stop words). Normalizing letters	BiGRU = 91% Acc.
[37]	2020	Machine Learning/Deep Learning	12 machine learning models and 2 deep learning models (RNN, CNN)	Arabic	Collection of 20,000 tweets and YouTube, Facebook, Instagram comments	Removing non-Arabic characters, emojis, URLs	RNN = 98.7% Acc.
[38]	2020	Machine Learning	SVM	Arabic	Collection of 10,000 tweets	Data cleaning, symbol removal labeling, tokenization	SVM = 90.2% Acc.
[39]	2020	Machine Learning/Deep Learning	LSTM, XGBoost	Arabic	Collection of 10,000 tweets	Removing punctuation marks and non-Arabic characters.	LSTM = 96% Acc.
[40]	2020	Machine Learning	AraBERT	Arabic	Collection of 10,000 comments	Tokenization (removing user mentions, retweets, URLs, diacritics emojis and newlines), replacing underscore in hashtags with white spaces	AraBERT = 90.15% F1
[41]	2020	Deep Learning	RNN, CNN	Arabic	Collection of 9316 tweets	Removing non-Arabic characters, emojis, URLs	CNN = 83% Acc.
[42]	2021	Machine Learning	SVM, NB, DT, RF	Arabic	Collection of 3696 tweets	Removing (diacritics, punctuation, non-Arabic characters, emoticons, and stop words). Normalizing letters	RF = 91.3% Acc.

Table 1. *Cont.*

Rf.	Year	Techniques	Classifier	Language	Datasets	Preprocessing	Max-score
[43]	2021	Deep Learning	CNN, CNNLSTM, BiLSTM-CNN	Arabic	Collection of 23,678 comments	Removing (diacritics, punctuation, non-Arabic characters, emoticons, and stop words). Normalizing letters	BiLSTM = binary label: 80% ternary label: 74% Multi-label: 73%
[21]	2021	Deep Learning/Machine Learning	CNN, BERT ALBERT, RoBERTa, LR	English	Collection of 3843 tweets	Removing special characters, numbers, newlines, mention tags	(CNN + BERT-large-uncased + BERT-base-uncased + ALBERT-xxlarge-v2) = 85.5% Acc,
[23]	2021	Deep Learning/Machine Learning	LR, SVM CNN, LSTM BiLSTM XLM, mBERT BETO	Spanish	HaterNet dataset: collection of 6000 tweets HatEval dataset: collection of 6600 tweets	Normalizing URLs, emails, users' mentions, percent, money, time, date expressions and phone numbers.	BETO = 65.8% F1 (HaterNet) 75.5% F1 (HatEval).
[44]	2022	Deep Learning	RNN	Arabic	Collection of 4203 comments	Data cleaning, symbol removal labeling, tokenization	RNN = binary label: 99.73% ternary label: 95.38% Multi-label: 84.14% Acc.
[45]	2022	Deep Learning	CNN, CNN-LSTM, BiLSTM	Arabic	Collection of 15,050 comments	Data cleaning, symbol removal labeling, 39 enization	CNN-LSTM = 87.27% 83.65% F1, 83.89% P 83.46% R
[47]	2022	Machine Learning	GA- SVM GA- XGBoost	Arabic	Collection of 4505 tweets	Removing non-Arabic characters, emojid, URLs	SVM = 88% Acc.
[19]	2022	Deep Learning	CNN, LSTMs, BiLSTMs, LSTM, GRU, and CNN-LSTM.	Ethiopian	Collection of 42,100 tweets and Facebook comments	18 enization, normalizing stop words removal	Bi-LSTM = 91% F1
[20]	2022	Deep Learning/Machine Learning	CNN + GRU BERT HAN	Turkish	Collection of 18,316 news articles	Removog non-Turkish characters, numbers, URLs	BERT = 90.4% Acc.

Table 2. Publicly available Arabic language datasets.

Rf. of Study Used	Dataset	Source	Region	Size	Labels	Year	Rf. of Dataset
[29,41]	Religious hate speech dataset	X	Multinational	6136 Tweets	Hate (2762), not hate (3374)	2019	[29]
[29,43]	L-HSAB Hate speech and offensive language	X	Levantine	5846 Tweets	Normal (3650), offensive (1728), hate (468)	2019	[31]
[33]	T-HSAB Hate speech and offensive language	X	Tunisian	6075 Tweets	Normal (3834), offensive (1127), hate (1078)	2019	[33]
[35,38–40,43]	OSACT4 Hate speech and offensive language	X	Multinational	20,000 Tweets	Offensive (1900), not offensive (8100), hate (50), not hate (9950)	2020	[35]
[45]	Offensive language	YouTube	Multinational	14,202 Comments	Offensive (9349), not offensive (4853)	2018	[45]
[47]	ArCybC Offensive language, cyberbullying	X	Multinational	4505 Tweets	Offensive (1887), cyberbullying (1728), Normal (890)	2022	[47]

4. Materials and Methods

The system employs a two-layer approach based on various machine learning and deep learning models to detect and classify hate speech. The first layer is dedicated to detecting hate speech, while the second layer focuses on classifying the detected hate speech into specific categories. The process begins with the collection of a comprehensive text Arabic dataset that serves as the basis for both layers of the system. In the first layer, feature extraction techniques are applied to prepare the dataset for analysis. To ensure a balanced and robust evaluation, a stratified five-fold cross-validation is performed, where the model is trained on four folds and tested on the fifth over several iterations. Predictions are generated for each iteration, and the overall performance of the hate speech detection model is calculated by averaging the results of these iterations.

In the second layer, the predictions of the best performing detection model are used to filter out non-hate content, leaving only hate speech data for further classification. This filtered dataset is subjected to classification into specific hate speech categories, with stratified five-fold cross-validation ensuring consistent and accurate scoring. The performance metrics are measured iteratively, and the final average score represents the effectiveness of the system in distinguishing between different types of hate speech. By integrating these two layers, the system provides a comprehensive and structured solution for the detection and classification of hate speech. The proposed system is shown in Figure 2.

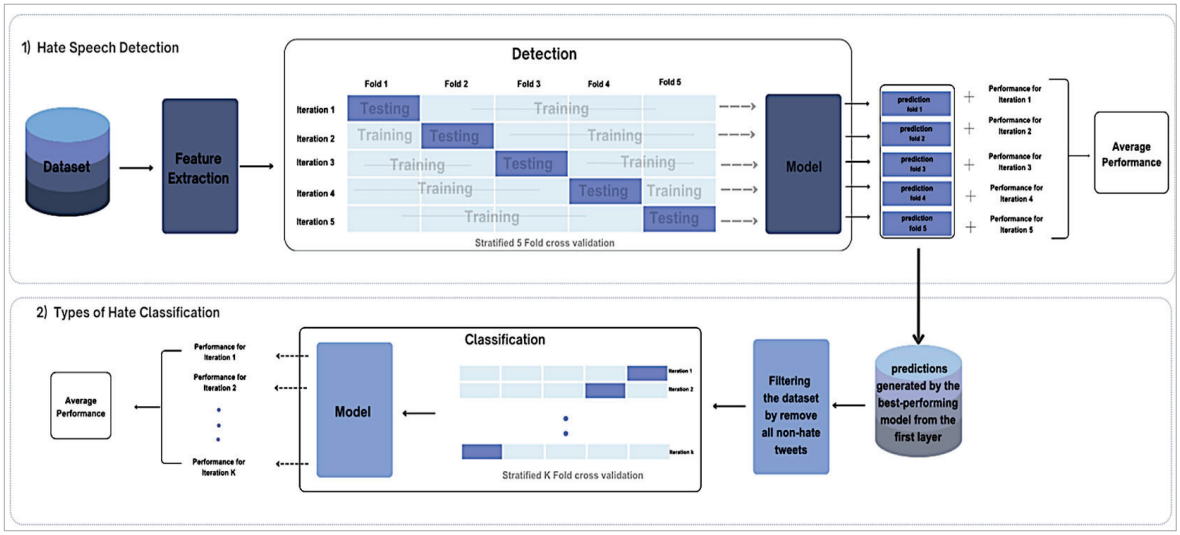


Figure 2. Proposed framework for hate speech detection and classification.

In order to test the performance of different methods and algorithms, we applied and training machine learning and deep learning algorithms for the detection and classification of Arabic hate speech. Outperforming all deep learning models, we noticed that CNN performed better when used in conjunction with pre-trained embeddings. The reasons for the decision of CNN are as follows. First, CNN's architecture to extract local features in textual data is highly beneficial in a morphologically rich language such as Arabic to detect hate speech. Second, the proposed model, CNN, is computationally efficient, since its nonsequential architecture does not include overheads arising from models like RNNs or LSTMs; thus, it is suitable for training with large datasets, such as Arabic tweet texts. Moreover, it is also seen in text classification and hate speech detection that the CNN outperforms sequences in short, context-free text because of its independence. Lastly, CNN is incorporated together with pre-trained embeddings, meaning that the localized feature extraction complements the global semantic knowledge incorporated in these representations.

N-grams and embedding of vectors from previous training were used in the study to extract features from raw data. N-grams were used to identify syntactic structures or sequences of terms and used for the detection of unambiguous hate speech content. Specifically, AraBERT and FastText covered the semantic aspects that the model required to identify other types of hate speech, such as idioms.

To deal with the nature of Arabic, the preprocessing comprised of diacritic removal, which lowers variability and normalization of giving words similar forms. Stemming was utilized to address issues of morphological differences, which counteracted noise and managed to keep words having the same root with the same status. These steps improved the data quality of the dataset and the model that was used in this study.

Three annotators, who are native Arabic speakers and majors in linguistics, were used to tag the dataset. To keep the labeling criteria consistent, annotators worked based on a set of instructions that highlighted aspects of an image that might be pertinent for a particular malware; in case of a disagreement between the annotators, the majority rule prevailed. Inter-annotator reliability was also assessed using Fleiss' kappa, which gave a substantial reliability coefficient of 0.78. As the annotator team was small, all the guidelines were prescribed carefully and there were objectives to measure the agreement among the annotators.

Because of the richness of the Arabic language in words and grammar, preprocessing entailed the removal of the diacritics, stemming and lemmatization to merge words that have similar stems but with different inflections. Particular emphasis was paid to colloquial utterances and epithets that are often used in social media platforms. This was to neutralize the manifest dialects from the gulf and Levantine regions in the data collection process. However, some regionalisms did not receive adequate representation in our study and warrant the collection of additional data for enhanced dialectal representation.

4.1. Experimental Dataset and Setup

To create the dataset for hate speech analysis, a structured approach was followed. Initially, relevant keywords representing different types of hate speech, such as religious, political and social hate speech, were identified using various sources, including specialized databases including HATEBASE [50] and an Arabic lexicon [30]. Ahmad et al. [51] underscore the critical role of corpus design and evaluation in Arabic hate speech detection, which aligns with our focus on constructing a dialectically balanced dataset. These keywords were then used to extract tweets from the social media platform X, using the X API integrated with the Tweepy library. After retrieving approximately 9700 tweets, a filtering process removed duplicates and irrelevant entries, resulting in 8341 tweets for analysis. Preprocessing involved cleansing, tokenization and normalization using tools such as PyArabic for Arabic text processing and NLTK for further text analysis. These steps ensured a high quality dataset that was suitable for annotation and subsequent classification. These steps produced an accurate dataset that was required for annotation and further classification. The preprocessing techniques were based on literature review on text pre-processing and feature extraction to use in modeling hate speech in the social media. Studies such as [18,38,40,44,45] show that there is lot of efficaciousness in diacritic stripping, stemming and tokenization, for example, while handling the challenges of Arabic language in computational processes. This radically minimizes the noise, density and superiority representation of features or configurations in the dataset that feeds machine learning algorithms.

Following the dataset creation and preprocessing, the annotation phase was critical for preparing the data for supervised machine learning. This process involved two key tasks. Initially, tweets were annotated as either "hate speech" or "not hate speech", providing a

binary classification. Tweets identified as hate speech were further categorized into one of three subtypes. To ensure quality and reliability, three native Arabic-speaking annotators were selected, based on their expertise and willingness to participate. Clear annotation guidelines were provided and disagreements in the binary classification were resolved by majority voting. The average text length of tweets in the dataset was approximately 22 words, and inter-annotator agreement was validated with a Fleiss' kappa score of 0.78, indicating substantial agreement. For the subcategories, tweets with complete disagreement among annotators were excluded, resulting in a final dataset of 7923 tweets, as shown in Figure 3. The reliability of these annotations was evaluated using Fleiss' kappa.

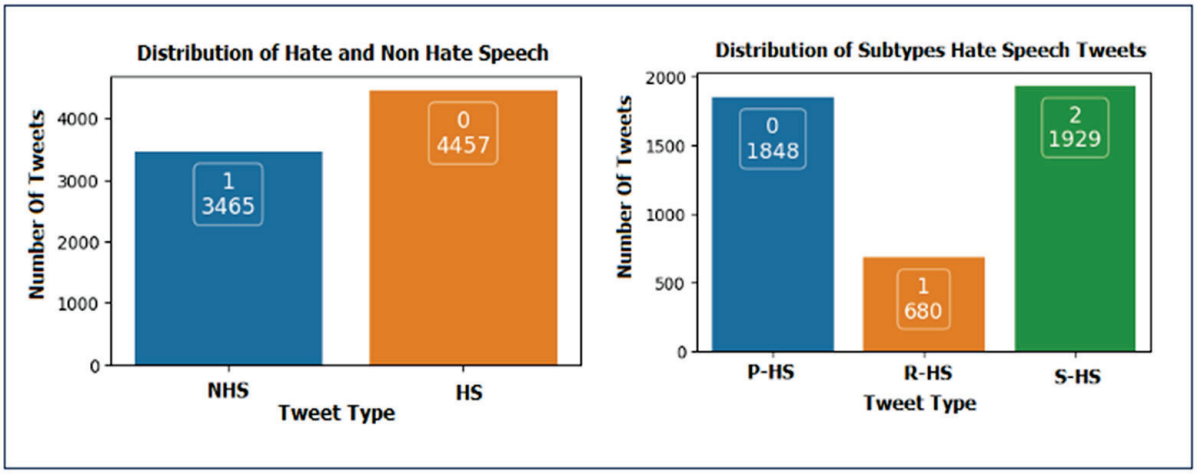


Figure 3. Distribution of tweets across dataset classes.

The experiments were conducted on macOS Monterey for initial tests and on Google Colab Pro with an A100 GPU for efficient model training. Python 3 was used in both setups to ensure smooth implementation and faster processing.

4.2. Data Representation (Feature Extraction)

Data representation plays a crucial role in preparing textual data for machine learning and deep learning models, as it transforms the raw textual information into a format that can be understood and effectively processed by these models. In this study, various techniques were applied to extract meaningful features from Arabic texts tailored to the specific requirements of machine learning and deep learning approaches;, these methods are:

- N-gram Feature Extraction: The text data were vectorized using the TF-IDF method, with configurations for word and character N-grams.
- Word N-grams: In a text, word N-grams store groups of words and show how these words relate to each other in context. To improve the feature extraction process, this study used word N-grams between 1 and 3. Using a range of 1 to 3 allows the model to analyze single words (unigrams), pairs of consecutive words (bigrams), and three-word sequences (trigrams). This approach allows the model to understand both the meanings of individual words and the relationship of words in short lines to each other.
- Character N-grams: In character N-grams, the study used characters that span up to 6 characters. This approach is very efficient for languages that have a rich morphological structure, such as Arabic, as it captures the morphological part of words. It also detects different spellings and common spelling errors that occur in digital communication.

- Word/Character N-grams: The study investigated a hybrid approach of word and character N-grams, where word N-grams from 1 to 3 were integrated with character N-grams from 1 to 6 to take advantage of both representations.
- Word Embeddings: For the deep learning models, the study used different methods to embed words. For CNN, RNN, LSTM and BiLSTM, we experimented with text representations by using trainable embeddings and pre-trained embeddings separately. More specifically, the pre-trained embeddings were generated from AraVec. AraVec is a pre-trained word embedding model that converts Arabic words into continuous vector representations. This model has been trained on large collections of Arabic texts, so that it can capture the complex linguistic patterns that characterize the Arabic language. The trainable embeddings, on the other hand, allowed the models to directly adapt the word representations to the dataset during the training process.

AraBERT and BiAraBERT were used in their pre-trained models for fine-tuning in the text classification task of Arabic tweet hate speech detection. The pre-training, which integrates both, provides a deep understanding of the structure of the language, while the fine-tuning adapts it to the subtleties of the dataset.

4.3. Evaluation and Validation

4.3.1. Stratified K-Fold Cross-Validation

To address the imbalance in the dataset, stratified K-fold cross-validation was utilized to evaluate the performance of the hate speech detection and classification models. This approach enhances traditional K-fold cross-validation by ensuring that each fold maintains the same class proportions as the original dataset, thereby preventing distortions caused by uneven distributions. The dataset is divided into K-folds, with K−1 folds used for training and one for testing in each iteration. This process was repeated K times, ensuring every fold serves as the test set once. For this study, K = 5 was selected, providing a balanced and robust assessment of the models. Figure 4 illustrates the class distribution across folds in the first (a) and second (b) layers of the hate speech classification system.

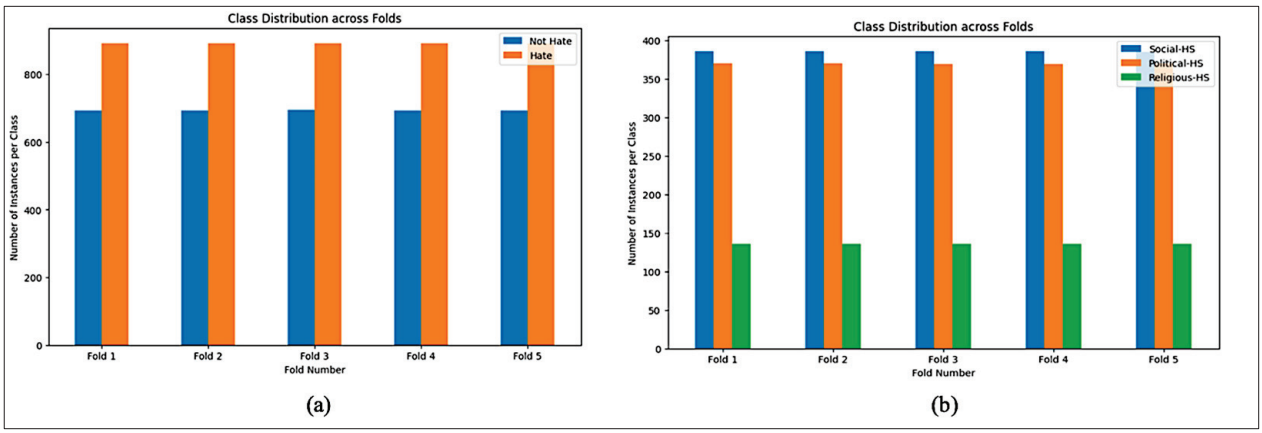


Figure 4. In panel (**a**), the distribution of "Hate" and "Not Hate" classes is shown for each of the 5 folds used during stratified K-fold cross-validation. Panel (**b**) depicts the distribution of hate speech categories—"Social-HS", "Political-HS", and "Religious-HS"—in the second layer across the same folds.

The method ensures that the class distributions are preserved using the following equations:

- Class proportion in the dataset

$$i = \frac{n_i}{N} \tag{1}$$

where:

- n_i is the number of samples of class i in the dataset.
- n is the total number of samples in the dataset.

- Class proportion in each fold:

$$\frac{n_{i.j}}{n_j} \approx \frac{n_i}{N} \tag{2}$$

where:

- $n_{i.j}$ is the number of samples of class i in fold j.
- n_j is the total number of samples in fold j.

By applying stratified K-fold cross-validation with K = 5, the models are evaluated on folds that reflect the original dataset's class distribution, resulting in a more accurate and comprehensive evaluation, particularly for imbalanced datasets.

4.3.2. Computational Complexity Analysis

The theoretical time complexity of the CNN architecture is estimated to be O(n) to process sequential text. On the other hand, models such as AraBERT, which are developed based on transformers, have a complexity of O(n^2), owing to their attention mechanisms. This trade-off shows the appropriateness of CNNs for real-time hate speech detection without denying the accuracy improvements provided by transformers.

4.3.3. Evaluation Metrics

Several metrics were employed to assess the effectiveness of the proposed model. These commonly used metrics enable comparisons to be made with other techniques in the literature. They include the precision, recall, F1 score and accuracy. Detailed definitions and descriptions of these metrics are provided in Table 3.

Table 3. Evaluation metrics used for experiments.

Evaluation Metrics	Equation	Description
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$	Accuracy measures the proportion of correctly classified instances out of the total instances. It is calculated as the ratio of the sum of true positives (TP) and true negatives (TN) to the total number of instances, which includes true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN)
Precision	$\frac{TP}{TP+FP}$	Precision measures the proportion of true positives (TP) out of all positive predictions (TP + FP)
Recall	$\frac{TP}{TP+FN}$	Recall evaluates the proportion of true positives (TP) out of all actual positives (TP + FN)
F1	$2 * \frac{Precision*Recall}{Precision*Recall}$	F1- score indicated as F1 is the average of precision and recall.

Class imbalance affected the model's accuracy and recall of political hate speech and other categories as the minority. It was evident from precision–recall curves that the model had to make a compromise, resulting in precision and recall values of the minority classes as follows: political hate speech values were 79% and 74%, while those of religious hate speech were 91% and 89%, respectively. Further improvement might be achieved by mitigating this imbalance, using methods such as oversampling or by cost-sensitive learning.

5. Experimental Results and Discussion

In this study, we used a comprehensive training framework to evaluate the proposed two-layer system. This system, as described earlier, consists of two main layers. The first layer focuses on hate speech detection, while the second layer is dedicated to predefined categories of hate speech. At each layer, we experimented with different machine learning and deep learning models in combination with different word representation methods to achieve optimal performance for each task. The second layer relies directly on the results of the first layer and uses the predictions of the best performing model of the first layer as input for the second layer. According to the study by Aladeemy et al. [52], diacritic normalization and lemmatization are still indispensable operations in preprocessing content written in Arabic for sentiment and hate speech analysis. This also operates in specific domains such as Mednini et al. [53], where the authors focused on a specific domain known as hate speech about brands.

5.1. First Layer: Systematic Detection of Hate Speech

The first experiment focused on detecting Arabic hate speech tweets. Table 4 presents the results obtained using various machine learning models, including the support vector machine (SVM), random forest (RF), naïve Bayes (NB), and logistic regression (LR). It can be seen from Table 4 that the highest performance for Arabic hate speech detection was achieved by LR with a combination of word- and char-Ngrams as features. This configuration achieved an accuracy of 77.81%, a precision of 77.58%, a recall of 77.14% and an F1 score of 77.29%. These results show that the integration of features at the word and character level significantly improves the performance of the model compared to using the individual features.

Table 4. Results obtained using various machine learning models, including support vector machine (SVM), random forest (RF), naïve Bayes (NB), and logistic regression (LR).

		Metrics			
Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LR	**WordNgrams**	**74.90**	**74.88**	**73.83**	**74.07**
	Char-Ngrams	**76.86**	**76.74**	**75.95**	**76.18**
	Word + Char-N-grams	**77.81**	**77.58**	**77.14**	**77.29**
SVM	Word-N-grams	74.49	74.16	74.26	75.02
	Char-N-grams	77.16	76.7	76.86	77.39
	Word + Char-N-grams	77.06	76.82	76.9	77.28
NB	Word-N-grams	76.24	76.89	74.69	75.06
	Char-N-grams	76.01	75.73	75.3	75.45
	Word + Char-N-grams	76.93	76.6	76.46	76.51
RF	Word-N-grams	72.86	72.71	71.7	71.91
	Char-N-grams	74.76	74.91	73.42	73.71
	Word + Char-N-grams	74.45	74.62	73.06	73.35

A repetition of the experiment with deep learning models showed a significant improvement in hate speech detection. As shown in Table 5, Figures 5–7, the highest performance was achieved by the convolutional neural network (CNN) with trainable embeddings, which achieved an accuracy of 91.95%, a precision of 92.59%, a recall of 93.36% and an F1 score of 92.96%. This result outperformed all other models, including recurrent architectures such as RNN, LSTM and BiLSTM, as well as models using pre-trained embeddings, such as AraVec, AraBERT and DistilBERT. These results highlight the effectiveness of CNNs, especially with trainable embeddings, in capturing complex patterns for detecting Arabic hate speech.

Table 5. A repetition of the experiment with different deep learning models for the first layer.

		Metrics			
Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
RNN	Trainble Embeddings	85.81	89.48	84.55	86.93
LSTM		90.09	91.22	91.16	91.19
BiLSTM		88.82	91.52	88.07	89.73
CNN		**91.95**	**92.59**	**93.36**	**92.96**
(a) Trainable Embedding					
RNN	AraVec pretrained Embeddings	79.77	84.97	77.83	81.24
LSTM		79.97	82.68	81.47	82.07
BiLSTM		79.68	82.02	81.87	81.93
CNN		**92.75**	**94.11**	**93.18**	**93.64**
Arabert	AraBERT pretrained Embeddings	91.67	91.78	91.28	91.49
Distilbert		80.37	80.25	79.66	79.88
(b) Pretrained Embedding					

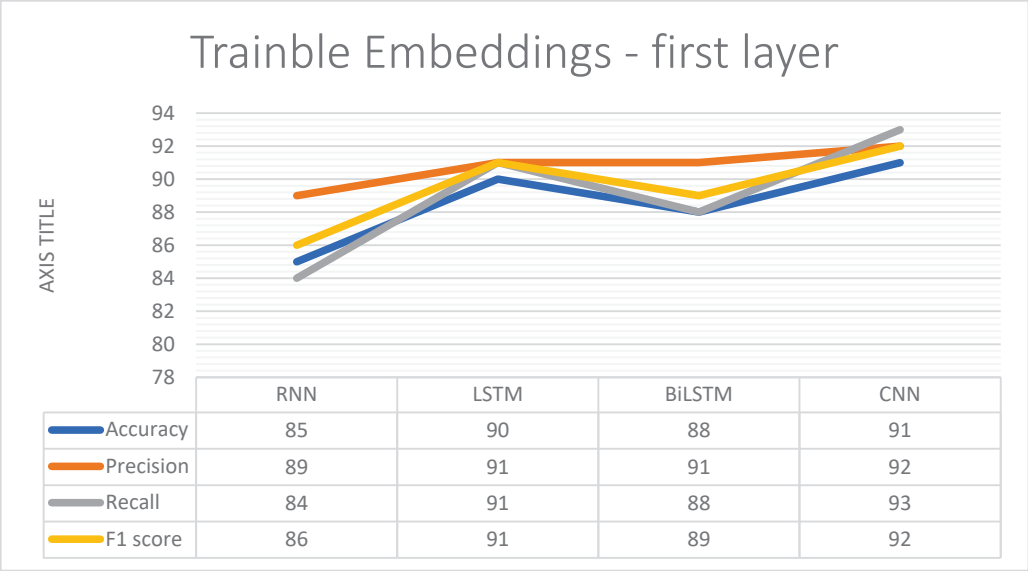


Figure 5. Precision–recall (PR) curve for a repetition of the experiment with different deep learning models for the first layer (trainable embeddings).

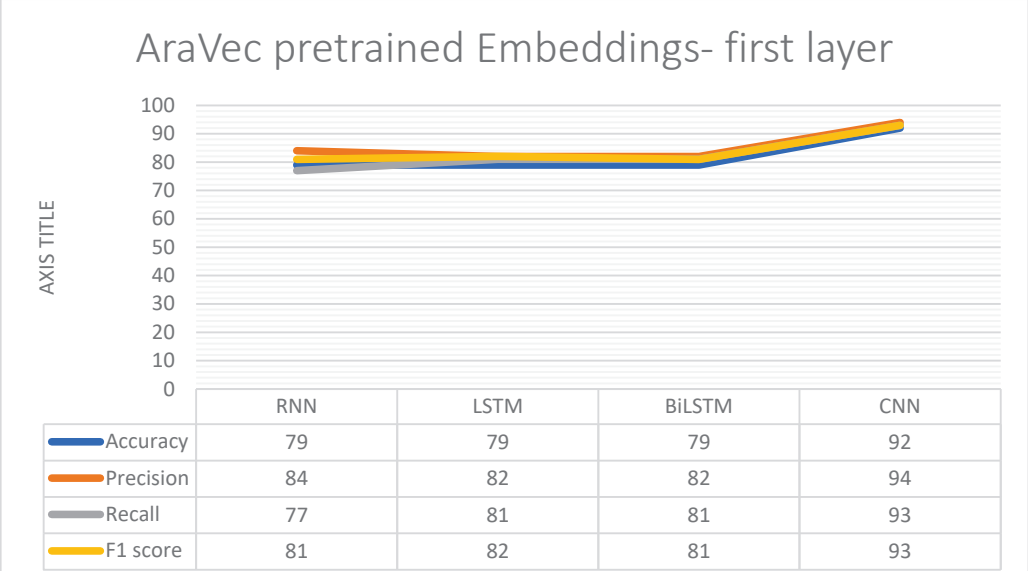


Figure 6. Precision–recall (PR) curve for a repetition of the experiment with different deep learning models for the first layer (pretrained embedding).

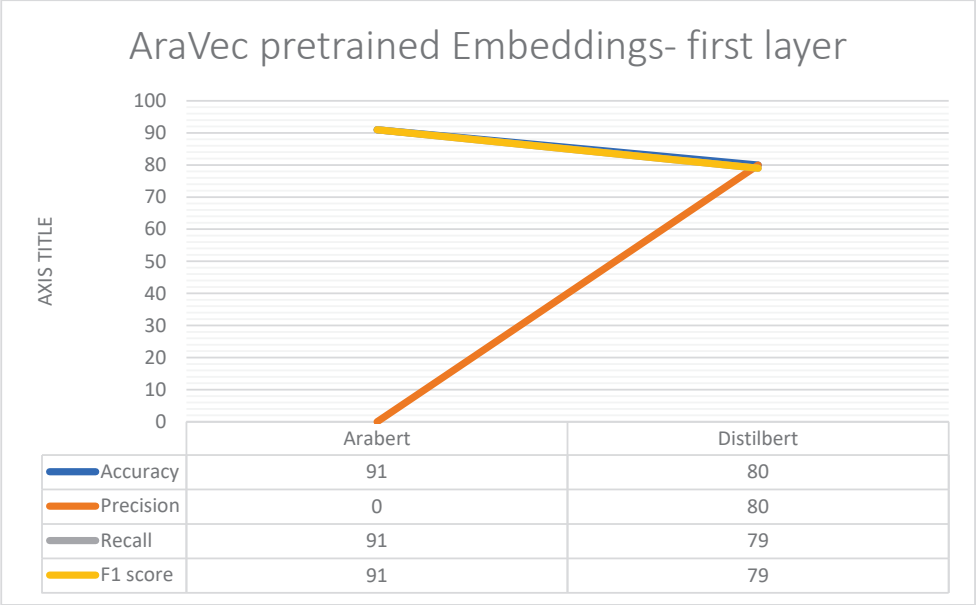


Figure 7. Precision–recall (PR) curve for a repetition of the experiment with different deep learning models for the first layer (AraBERT pretrained embeddings).

5.2. Second Layer: Systematic Classification of Hate Speech Types

In the second layer of our hate speech classification system, the predictions of the best performing model in the first layer, CNN, are used to classify hate speech into specific types. The predictions of each fold in the first layer are used as inputs for that layer, with non-hate speech texts filtered out to refine the results. This section evaluates the performance of different machine learning and deep learning models in classifying these types of hate speech, using different feature representations to improve the accuracy and effectiveness of the classification process. As shown in Table 6, Figures 8–10, logistic regression (LR) achieved the best performance, with an accuracy of 75.58% and an F1 score of 72.31% using a combination of word and character N-grams, demonstrating the effectiveness of this feature set in capturing nuanced patterns in hate speech. The support vector machine (SVM) also performed well, achieving 73.99% accuracy and a 72.13% F1 score with the same features, showcasing its robustness. In contrast, naive Bayes (NB) struggled with word N-grams alone, achieving an F1 score of just 53.19%, but improved with character N-grams, reaching 74.13% accuracy and a 68.33% F1 score. Overall, the results highlight the importance of combining word and character N-grams to enhance model performance, with LR and SVM emerging as the most reliable models for hate speech classification.

Table 6. Evaluation of the performance of different machine learning and deep learning models in classifying types of hate speech, using different feature representations to improve the accuracy and effectiveness of the classification process.

		Metrics			
Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LR	**Word-N-grams**	**71.1**	**74.39**	**62.81**	**64.84**
	Char-N-grams	**74.91**	**77.11**	**69.04**	**71.35**
	Word + Char-N-grams	**75.58**	**77.61**	**70.04**	**72.31**
SVM	Word-N-grams	72.22	74,34	66.59	68.83
	Char-N-grams	75.09	76.21	70.9	72.76
	Word + Char-N-grams	73.99	74.15	70.8	72.13
NB	Word-N-grams	68.02	75.76	55.09	53.19
	Char-N-grams	72.58	78.04	63.43	65.38
	Word + Char-N-grams	74.13	77.89	66.07	68.33
RF	Word-N-grams	69.77	73.32	62.55	64.76
	Char-N-grams	71.05	73.28	64.38	66.55
	Word + Char-N-grams	71.66	74.14	65.27	67.52

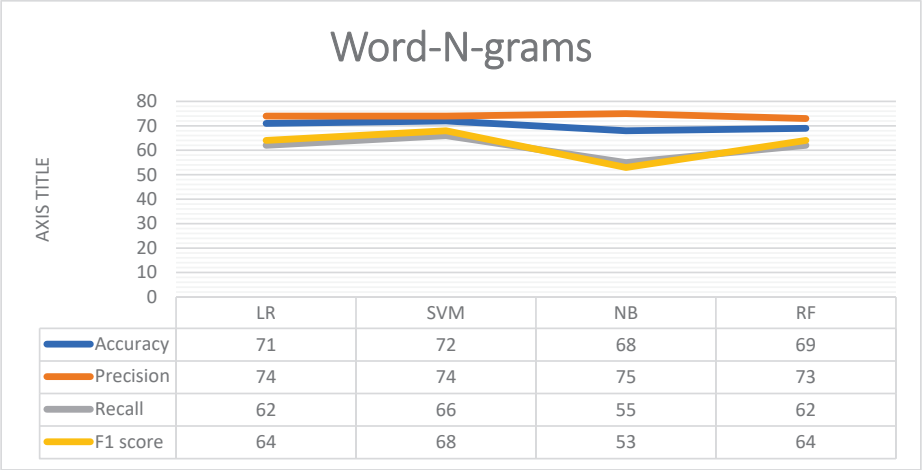


Figure 8. Evaluation of the performance of different machine learning and deep learning models (Word-N-grams feature).

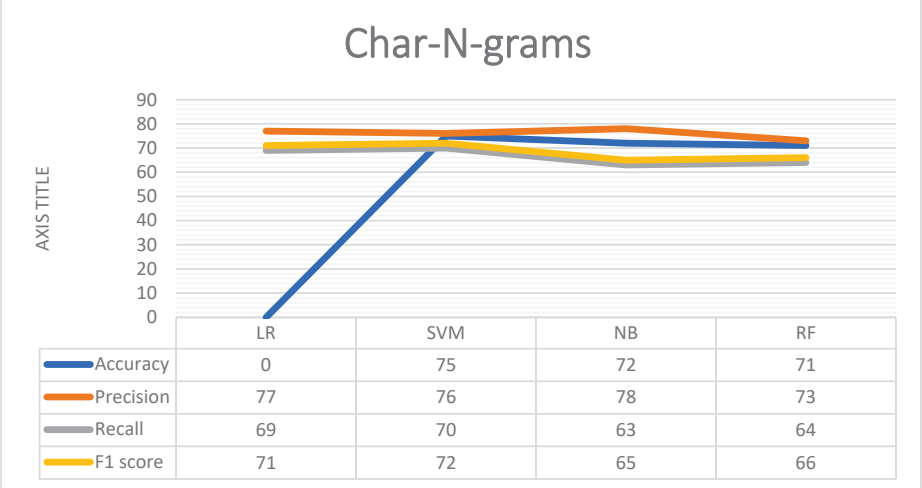


Figure 9. Evaluation of the performance of different machine learning and deep learning models (Char-N-grams feature).

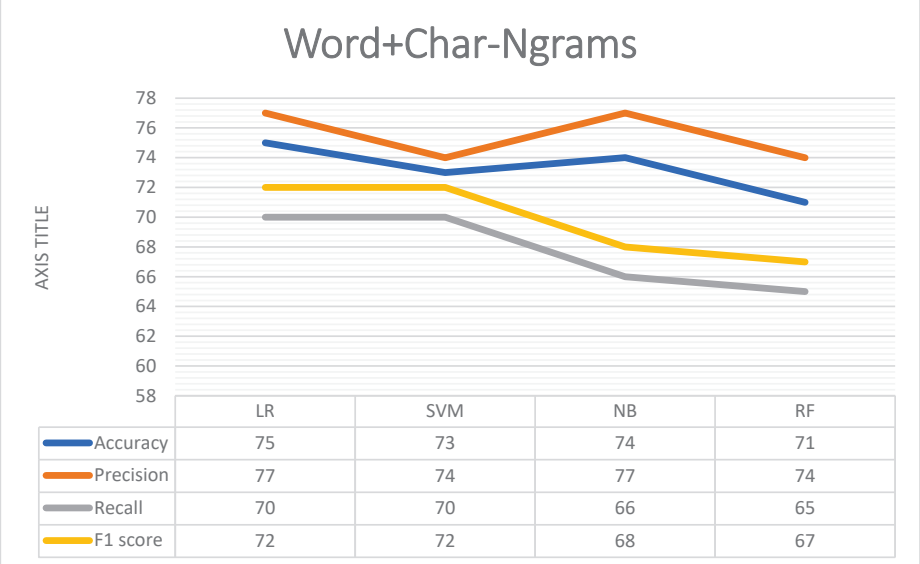


Figure 10. Evaluation of the performance of different machine learning and deep learning models (Word + Char-N-grams feature).

The evaluation results in Table 7, Figures 11–13 illustrate our experiments with deep learning models for hate speech classification and show that CNN is the most effective model. CNN achieved the highest accuracy of 93.23% with trainable embeddings and 93.33% with pre-trained embeddings and F1-scores of 91.93% and 92.36%, respectively. These results highlight CNN's ability to effectively capture local text features and patterns using convolutional filters and demonstrate its adaptability and consistency across different embedding types. AraBERT also performed well, particularly with pre-trained embeddings, achieving an accuracy of 84.29% and an F1 score of 84.15%, highlighting its strength in processing nuanced speech and contextual information. Although AraBERT's performance was slightly lower than CNN's, it highlights the importance of contextual embeddings for hate speech detection. Overall, CNN proves to be the most reliable model in this study.

Table 7. A repetition of the experiment with different deep learning models for the second layer.

		Metrics			
Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
RNN	Trainable Embeddings	85.13	84.14	81.33	82.26
LSTM		79.92	71.85	70.61	68.79
BiLSTM		83.27	72.33	72.81	71.84
CNN		**93.23**	**93.04**	**91.39**	**91.93**
(a) Trainable Embedding					
RNN	AraVec pretrained Embeddings	80.39	79.52	79.35	79.38
LSTM		80.68	80.14	80.00	80.01
BiLSTM		78.06	77.00	77.05	76.79
CNN		**93.33**	**93.33**	**91.95**	**92.36**
Arabert	AraBERT pretrained Embeddings	84.29	84.85	83.59	84.15
Distilbert		78.68	77.57	77.18	77.24
(b) Pretrained Embedding					

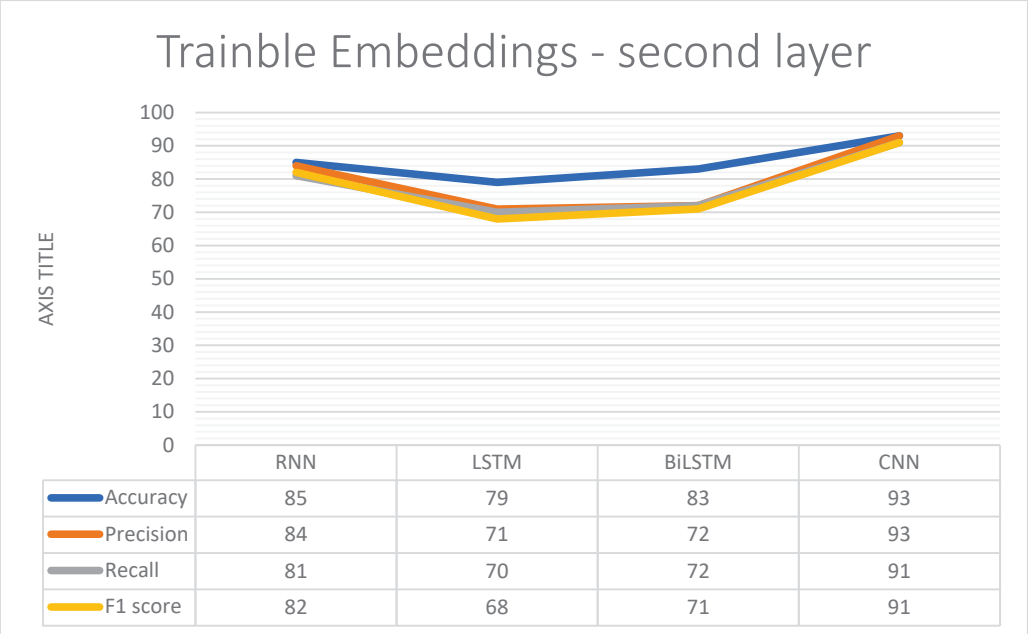


Figure 11. Precision–recall (PR) curve for a repetition of the experiment with different deep learning models for the second layer (AraBERT pretrained embeddings).

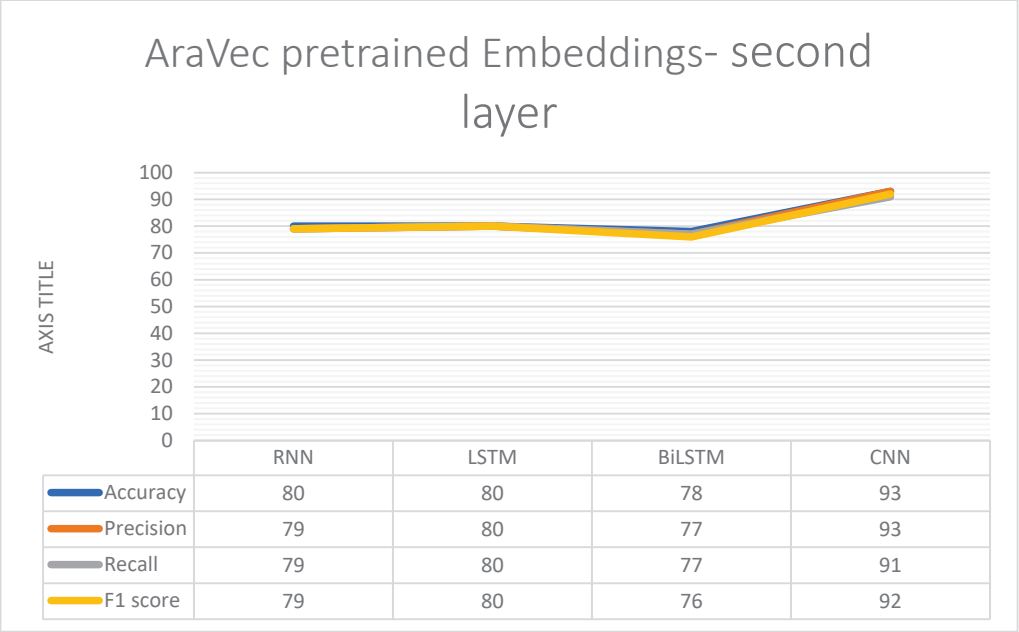


Figure 12. Precision–recall (PR) curve for a repetition of the experiment with different deep learning models for the second layer (AraBERT pretrained embeddings).

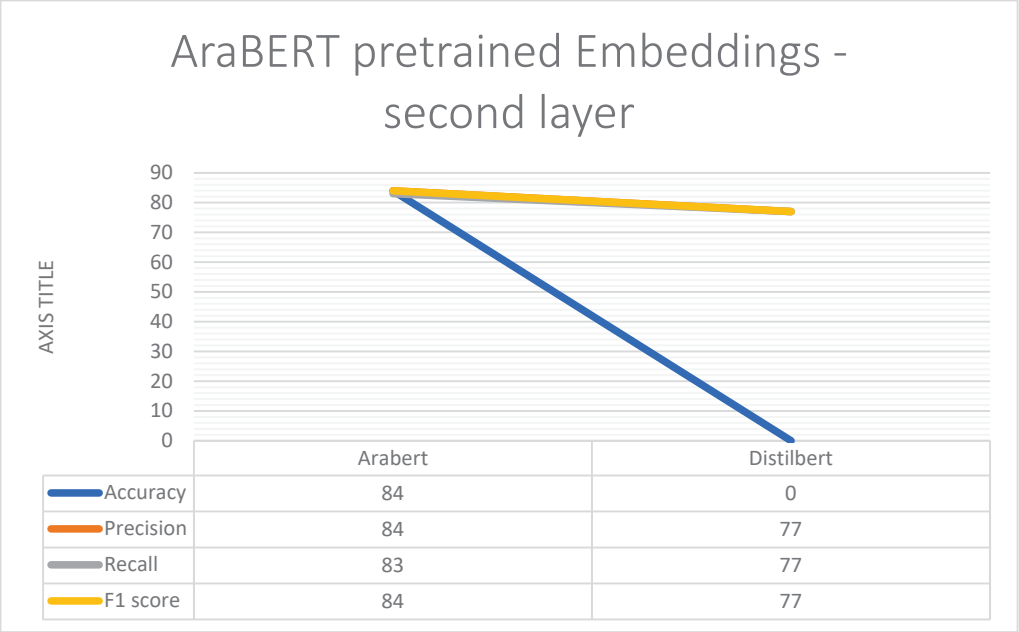


Figure 13. Precision–recall (PR) curve for a repetition of the experiment with different deep learning models for the second layer (AraBERT pretrained embeddings).

5.3. Confidence Intervals and Statistical Testing

Confidence intervals for accuracy and F1 scores were calculated as shown in Table 8. Paired *t*-tests demonstrated statistically significant performance differences between CNN and AraBERT ($p < 0.01$).

Table 8. Confidence intervals for accuracy and F1 scores for CNN.

Model	Metric	Mean (%)	95% CI (%)
CNN	Accuracy	92.0	[91.5, 92.5]
CNN	F1 Score	93.0	[92.4, 93.6]

5.4. Benchmarking Against State-of-the-Art Models

To rigorously evaluate the performance and generalizability of our proposed framework, we conducted comprehensive benchmarking experiments on two external datasets: L-HSAB and OSACT4. The purpose of selecting these datasets was to cover a wide range of different linguistic features and areas and, therefore, provide a rather challenging test for the model proposed herein. The findings of these experiments are shown in Table 9.

Table 9. Confidence intervals for accuracy and F1 scores for CNN.

Dataset	Model	Accuracy (%)	F1 Score (%)
L-HSAB	CNN	89.5	88.3
L-HSAB	AraBERT	91.8	90.7
OSACT4	CNN	87.4	85.0
OSACT4	AraBERT	90.2	88.5

5.5. Discussion

The evaluation results show that the proposed approach, based on CNN, performed better than other types of models in identifying and classifying Arabic hate speech. The improved performance was largely due to the use of pre-trained embeddings, reaching 92% accuracy in detecting hate speech and 93% accuracy in classifying it into the three classes of hate speech. This further highlight that CNN is better suited for extracting local features from text and that it benefits from the context of pre-trained embeddings. However, an examination of the findings reveals some errors in specific areas, as well as several limitations that need to be discussed.

Even though CNN showed promising results, it was compared to transformers such as AraBERT. Specifically, transformers outperform CNNs in modelling long-range dependencies, and the agreement ratio is 89%, which is slightly lower than CNN's 92%. Notably, fine-tuned Mamba was fairly equal in multilingual hate speech detection to the others, although Mamba demanded significant fine-tuning for the specificity of the Arabic language. These comparisons show that the proposed CNN achieves a good trade-off between performance and required computation, especially in the case of short texts such as tweets.

In terms of efficiency, this study showed that the framework had reliable performance, with its computational complexity ideal for practical use. The string matching takes only a few seconds when working with the dataset of 10,000 tweets, which suggests that the method can be useful for monitoring in real time. It achieves lower computational costs than transformers, making the model suitable for applications with limited computational resources.

This makes the identification of false positives a moot point; nevertheless, it is ethically questionable. Misclassifications can negatively impact the interactions on the platform and the trust that users put in it by silencing genuine speech. This is a problem that has to be solved by designing AI systems that provide a way to inspect content that AI has deemed problematic, and by implementing bias-detection features that would lower the likelihood of missteps made by AI. The conclusions of the present study align with the challenges and solutions suggested by Abdelsamie et al. [54] about the difficulty of dealing with abusive language in Arabic social media.

5.5.1. Error Analysis

One general problem faced by the model was that it was unable to correctly classify sarcastic or implicit hate speech. Of the tweets containing hate, 15% were misclassified because the language used sarcasm or indirectness. In comparison to having obvious, easily recognizable rude words, implicit hate speech may contain sarcasm, puns or references

to a specific culture, without necessarily having to contain Arabic words. For example, a seemingly positive comment, such as "We understand your concern", may be a taunt to a target group that is designed to hurt. The second major source of misclassification arose from dialectal differences within the Arabic language. Although Standard Arabic is the basis for the majority of standardized educational representations, the Gulf, Levantine, and Maghrebi varieties that have infiltrated social media do not differ significantly in vocabulary, grammar and usage. The model was not very effective about 12% of the time, mainly due to an inability to understand colloquial or regional expressions. This limitation suggests that there is an opportunity to develop dialect-specific datasets, or at least embeddings, to enhance the study of Arabic hate speech in its multicultural aspects.

Progression performance was also affected by the class imbalance. The "religious hate speech" class had quite high precision and recall values, of 0.91 and 0.89 respectively, for classifying the posts accurately, while the "political hate speech "class produced lower values of precision (0.79) and recall (0.74) for classification of the posts. This can primarily be attributed to the limited inclusion of the political hate speech in the dataset, in addition to the generalization capability of the model.

The majority of misclassifications occurred in samples of hate speech and sarcasm, when no straightforward marker of offensive content was present. It was found that these cases contributed nearly 15% of false negatives. Tweets containing landmarks also posed problems, while the dialect-specific expressions turned out to be perhaps the hardest, as embeddings derived mainly from Modern Standard Arabic did not code for regional differences. Including dialect-specific data or embeddings could help alleviate such errors.

5.5.2. Limitations of the Chosen Model

On the positive side, CNN architecture showed many advantages; on the negative side, however, certain disadvantages were detected. First, there is a practical problem of training embeddings, which depend on data from other sources and, therefore, may not necessarily reflect the Arabic hate speech in informal or emerging contexts. For instance, models mentioned in this paper, such as AraBERT, can be trained for this purpose but are not originally optimized for the specific task of hate speech detection.

Moreover, while CNN is stronger in local features within text, these same characteristics reduce its ability to understand broader, more complex contextual dependencies. In the case of longer tweets that contain more than one idea or when the hate speech is intricately woven within the text message, sometimes the CNN model barely got it right, because it was missing specific contextual information. In similar circumstances, CNN could be outperformed by transformer-based models such as AraBERT because they incorporate attention representations directed towards global and contextual communication.

One of the limitations involved is that of scalability of the detection system, which must be transposable at any time to accommodate for any type of instance. Although CNN offered high and efficient output during the research experiments, implementing this model in a large-scale real-time architecture can be difficult. Furthermore, handling new embeddings and managing streams of millions of tweets per day would require more enhancement and resources.

Last, the research focus was limited to text-based discrimination; other formats, ranging from images, videos or even multimedia content, as seen on platforms such as X (formerly Twitter) were not considered. This might have restricted the model's generalizability, as much of the hate speech that occurs in different social media platforms contains other forms of media that escalate the effect of hate speech.

5.5.3. Recommendations and Future Directions

To overcome the problems mentioned above, further research should focus on improved preprocessing strategies that are specific to the issues of dialectal variations and implicit hate speech. Using domain-specific lexicons in hate speech, tuning the embeddings on hate speech-relevant corpora or using different embeddings for different Arabic dialects could minimize the misclassification errors emerging from dialectal differences. Another approach could be to introduce other models using transformers such as AraBERT or utilize hybrid models that encase both CNN and transformer layers to approach the problem of CNN, which does not capture long or complex texts' global context. These models penetrate better in tasks that necessitate a better understanding of textual relations and may generate a good synergy with CNNs that are optimized for local feature extraction.

Concerning class imbalance, by using oversampling or improving cost-sensitive learning, the performance of the model for sorting underrepresented types of hate speech, such as political speech, can be improved. This is likely to result in a more uniform performance of the exemplary schools at all classes. Third, extending the model for the analysis to several modalities, and the combination of textual data with image or video processing, would greatly improve the model's reliability and versatility. As mentioned earlier, many hate speech messages on social media are image dependent, and it is thus necessary to incorporate such data.

6. Conclusions

This study presents an unpublished two-tier framework for Arabic hate speech identification and categorization, accompanied by limitations such as Arabic language richness and scarcity of datasets. Through their combination with the pre-trained embeddings, the CNNs provide high detection and classification rates. Future work focuses on working with transformer models to improve the hate speech detection and introduce features from other modalities to increase the protection of users on social media platforms.

This research processes a sentiment analysis or a similar task on texts of small sizes, such as tweets. Convolutional neural networks (CNNs) are employed due to their strengths, which include computational efficiency characteristics for working with large amounts of short texts and their ability to learn local patterns and features within the short sequences, which is especially important when analyzing the (tweet) texts. In addition, CNNs are especially effective when there are hardware and computational constraints in the environment.

A limitation of this current CNN approach is mentioned in the manuscript. This includes inadequate capabilities for the identification of long-range dependencies within the text and probable difficulties in processing syntactic features of the text in general. To overcome these limitations, the researchers intend to extrapolate from CNNs to examine more useful combinations of other architectures. This will comprise integrating CNN's with recurrent neural network (RNNs) or long short-term memory (LSTMS). CNNs can learn local features, and RNNs or LSTMs can learn long-term dependencies and sequential information better than can CNN. Moreover, if we integrate attention mechanisms into the model, the model will be able to pay more attention to the specific parts of the text and obtain better performance of the model, as well as better computational interpretability.

In summary, this research study employs CNNs on a task related to short text data, meaning that the data are in the form of tweets. The researchers understand and foresee the disadvantages of CNNs and are keen to incorporate a hybrid model to improve the models' shortcomings.

This study underscores the importance of detecting and classifying hate speech on OSNs such as X to ensure user safety and combat the spread of harmful content. This

study has made significant contributions by compiling a comprehensive Arabic dataset from X, providing a valuable resource for the analysis of hate speech. Furthermore, the development of a novel two-layer framework that integrates both machine learning and deep learning techniques, along with various feature and embedding methods, has proven effective in detecting and classifying hate speech into specific categories, such as religious, political, and social.

Future work aims to enhance the framework by utilizing unexploited dataset features such as emojis and images, improving natural language processing for Arabic dialects and extending the approach to a unified model for simultaneous hate speech detection and classification. These advancements promise to address the challenges in automatic hate speech detection and further bolster user protection.

Author Contributions: Data curation, S.A. (Sadeem Alrasheed); formal analysis, S.A. (Sadeem Alrasheed); methodology, S.A. (Sadeem Alrasheed); resources, S.A. (Sadeem Alrasheed); software, S.A. (Sadeem Alrasheed); writing—original draft preparation, S.A. (Sadeem Alrasheed); investigation, S.A. (Suliman Aladhadh) and A.A.; project administration, S.A. (Suliman Aladhadh) and A.A.; supervision, S.A. (Suliman Aladhadh) and A.A.; validation, S.A. (Suliman Aladhadh) and A.A.; writing—review and editing, S.A. (Suliman Aladhadh) and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge Qassim University, represented by the Deanship of Graduate Studies and Scientific Research, on the financial support for this research under the number (QU-J-PG-2-2025 - 54284) during the academic year 1446 AH/2024 AD.

Data Availability Statement: Authors have created a GitHub repository for this thesis, which contains all the code, datasets and documentation for the project titled "Protecting Intellectual Security Through Hate Speech Detection Using Artificial Intelligence Approaches". GitHub Repository https://github.com/Sadeem1Alrasheed/Arabic-Hate-Speech-Detection-and-Classification, accessed on 28 February 2025).

Conflicts of Interest: The authors declare that they have no conflict of interest to report regarding the present study.

References

1. Robinson, L.; Smith, M. Social Media and Mental Health: Social Media Addiction. HelpGuide.org. Available online: https://www.helpguide.org/mental-health/wellbeing/social-media-and-mental-health (accessed on 15 April 2024).
2. Maarouf, A.; Pröllochs, N.; Feuerriegel, S. The Virality of Hate Speech on Social Media. *Proc. ACM Hum. Comput. Interact.* **2024**, *8*, 1–22. [CrossRef]
3. Al-Hassan, A.; Al-Dossari, H. Detection of Hate Speech in Arabic Tweets Using Deep Learning. *Multimed. Syst.* **2021**, *28*, 1963–1974. [CrossRef]
4. Alnazzawi, N. Using Twitter to Detect Hate Crimes and Their Motivations. *HateMotiv Corpus. Data* **2022**, *7*, 69. [CrossRef]
5. Mateusz, B. The Impact of Social Media on Modern Communication: How Platforms like Facebook and Twitter Influence the Way We Connect with Each Other. aithor.com. Available online: https://aithor.com/essay-examples/the-impact-of-social-media-on-modern-communication-how-platforms-like-facebook-and-twitter-influence-the-way-we-connect-with-each-other (accessed on 9 April 2024).
6. Meikle, G. *Social Media*; Routledge: London, UK, 2016. [CrossRef]
7. Rehman, A.; Khan, M.; Abbas, A.; Javed, M.; Abbas, M.; Hussain, M.; Ul-Allah, S. Evaluation of genetic variability and heritability of wheat genotypes under late sowing effects. *Biol. Clin. Sci. Res. J.* **2023**, *2023*, 268. [CrossRef]
8. Alhazmi, A.; Mahmud, R.; Idris, N.; Abo, M.E.M.; Eke, C. A systematic literature review of hate speech identification on Arabic Twitter data: Research challenges and future directions. *PeerJ Comput. Sci.* **2024**, *10*, e1966. [CrossRef]
9. Poletto, F.; Basile, V.; Sanguinetti, M.; Bosco, C.; Patti, V. Resources and Benchmark Corpora for Hate Speech Detection: A Systematic Review. *Lang. Resour. Eval.* **2020**, *55*, 477–523. [CrossRef]
10. Laub, Z. Hate Speech on Social Media: Global Comparisons. Council on Foreign Relations. Available online: https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons (accessed on 12 January 2024).

11. Das, S. Twitter Fails to Delete 99% of Racist Tweets Aimed at Footballers in Run-Up to World Cup. The Guardian. Available online: https://www.theguardian.com/technology/2022/nov/20/twitter-fails-to-delete-99-of-racist-tweets-aimed-at-footballers-in-run-up-to-world-cup (accessed on 1 November 2023).
12. Faris, H.; Ibrahim, A.; Habib, M.; Castillo, P. Hate Speech Detection Using Word Embedding and Deep Learning in the Arabic Language Context. In Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods ICPRAM, Valletta, Malta, 22–24 February 2020; SciTePress: Setúbal, Portugal, 2020; Volume 1, pp. 453–460. [CrossRef]
13. Radcliffe, D.; Abuhmaid, H.; Mahliaire, N. *Social Media in the Middle East 2022: A Year in Review*; SSRN Electronic Journal: Rochester, NY, USA, 2023. [CrossRef]
14. Cohen-Almagor, R. Fighting Hate and Bigotry on the Internet. Ssrn.com. Available online: https://ssrn.com/abstract=1916552 (accessed on 9 May 2024).
15. Jahan, M.S.; Oussalah, M. A systematic review of hate speech automatic detection using natural language processing. *Neurocomputing* **2023**, *546*, 126232. [CrossRef]
16. Chaudhary, M.; Saxena, C.; Meng, H. Countering online hate speech: An nlp perspective. *arXiv* **2021**, arXiv:2109.02941.
17. Brown, A.; Sinclair, A. *The Politics of Hate Speech Laws*; Routledge: London, UK, 2019.
18. Díaz-Faes, D.A.; Pereda, N. Is there such a thing as a hate crime paradigm? An integrative review of bias-motivated violent victimization and offending, its effects and underlying mechanisms. *Trauma Violence Abus.* **2022**, *23*, 938–952. [CrossRef]
19. Moon, R. *Putting Faith in Hate*; Cambridge University Press: Cambridge, UK, 2018.
20. Ganfure, G.O. Comparative Analysis of Deep Learning Based Afaan Oromo Hate Speech Detection. *J. Big Data* **2022**, *9*, 76. [CrossRef]
21. Hüsünbeyi, Z.M.; Akar, D.; Özgür, A. Identifying Hate Speech Using Neural Networks and Discourse Analysis Techniques. In Proceedings of the First Workshop on Language Technology and Resources for a Fair, Inclusive, and Safe Society within the 13th Language Resources and Evaluation Conference, Marseille, France, 20–25 June 2022; pp. 32–41.
22. Jahan, M.S.; Beddiar, D.; Oussalah, M.; Arhab, N.; Bounab, Y. Hate and Offensive Language Detection Using BERT for English Subtask A. In Proceedings of the Working Notes of FIRE 2021-Forum for Information Retrieval Evaluation, Gandhinagar, India, 13–17 December 2021; RWTH Aachen University: Aachen, Germany, 2021.
23. Srivastava, A.; Hasan, M.; Yagnik, B.; Walambe, R.; Kotecha, K. Role of Artificial Intelligence in Detection of Hateful Speech for Hinglish Data on Social Media. In *Applications of Artificial Intelligence and Machine Learning: Select Proceedings of ICAAAIML 2020*; Springer Singapore: Singapore, 2021; pp. 83–95. [CrossRef]
24. Aftan, S.; Shah, H. Using the AraBERT Model for Customer Satisfaction Classification of Telecom Sectors in Saudi Arabia. *Brain Sci.* **2023**, *13*, 147. [CrossRef]
25. Koshiry, A.M.E.; Eliwa, E.H.I.; Abd El-Hafeez, T.; Omar, A. Arabic Toxic Tweet Classification: Leveraging the AraBERT Model. *Big Data Cogn. Comput.* **2023**, *7*, 170. [CrossRef]
26. Antoun, W.; Baly, F.; HAJJ, H. Arabert: Transformer-based model for arabic language understanding. *arXiv* **2020**, arXiv:2003.00104.
27. Pires, T. How multilingual is multilingual BERT. *arXiv* **2019**, arXiv:1906.01502.
28. Wu, S.; Dredze, M. Are all languages created equal in multilingual BERT? *arXiv* **2020**, arXiv:2005.09093.
29. Plaza-del-Arco, F.M.; Molina-González, M.D.; Ureña-López, L.A.; Martín-Valdivia, M.T. Comparing Pre-Trained Language Models for Spanish Hate Speech Detection. *Expert Syst. Appl.* **2021**, *166*, 114120. [CrossRef]
30. Albadi, N.; Kurdi, M.; Mishra, S. Are they Our Brothers? Analysis and Detection of Religious Hate Speech in the Arabic Twittersphere. In Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, 28–31 August 2018. [CrossRef]
31. Albadi, N. nuhaalbadi/Arabic_hatespeech. GitHub. Available online: https://github.com/nuhaalbadi/Arabic_hatespeech (accessed on 10 May 2024).
32. Mulki, H.; Haddad, H.; Bechikh Ali, C.; Alshabani, H. L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language. In Proceedings of the Third Workshop on Abusive Language Online, Florence, Italy, 1 August 2019; pp. 111–118. [CrossRef]
33. Hala-Mulki. Hala-Mulki/L-HSAB-First-Arabic-Levantine-HateSpeech-Dataset. GitHub. Available online: https://github.com/Hala-Mulki/L-HSAB-First-Arabic-Levantine-HateSpeech-Dataset (accessed on 10 May 2024).
34. Haddad, H.; Mulki, H.; Oueslati, A. T-HSAB: A Tunisian Hate Speech and Abusive Dataset. In *Communications in Computer and Information Science*; Springer International Publishing: Cham, Switzerland, 2019; pp. 251–263. [CrossRef]
35. Hala-Mulki. Hala-Mulki/T-HSAB-A-Tunisian-Hate-Speech-and-Abusive-Dataset. GitHub. Available online: https://github.com/Hala-Mulki/T-HSAB-A-Tunisian-Hate-Speech-and-Abusive-Dataset (accessed on 24 May 2024).
36. Haddad, B.; Zoher, O.; Anas, A.-A.; Ghneim, N. Arabic Offensive Language Detection with Attention-Based Deep Neural Networks. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, Marseille, France, 11–16 May 2020; pp. 76–81.

37. The 4th Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT4). edinburghnlp.inf.ed.ac.uk. Available online: https://edinburghnlp.inf.ed.ac.uk/workshops/OSACT4/ (accessed on 3 June 2024).
38. Omar, A.; Mahmoud, T.M.; Abd-El-Hafeez, T. Comparative Performance of Machine Learning and Deep Learning Algorithms for Arabic Hate Speech Detection in OSNs. In Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020), Cham, Switzerland, 24 March 2020; pp. 247–257. [CrossRef]
39. Husain, F. OSACT4 Shared Task on Offensive Language Detection: Intensive Preprocessing-Based Approach. *arXiv* **2020**, arXiv:2005.07297.
40. Alharbi, A.I.; Lee, M. Combining Character and Word Embeddings for the Detection of Offensive Language in Arabic. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, Marseille, France, 11–16 May 2020; pp. 91–96.
41. Djandji, M.; Baly, F.; Antoun, W.; Hajj, H. Multi-Task Learning Using AraBert for Offensive Language Detection. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, Marseille, France, 11–16 May 2020; pp. 97–101.
42. Alshaalan, R.; Al-Khalifa, H. Hate Speech Detection in Saudi Twittersphere: A Deep Learning Approach. In Proceedings of the Fifth Arabic Natural Language Processing Workshop, Barcelona, Spain, 12 December 2020; pp. 12–23.
43. Aljarah, I.; Habib, M.; Hijazi, N.; Faris, H.; Qaddoura, R.; Hammo, B.; Abushariah, M.; Alfawareh, M. Intelligent Detection of Hate Speech in Arabic Social Network: A Machine Learning Approach. *J. Inf. Sci.* **2020**, *47*, 483–501. [CrossRef]
44. Duwairi, R.; Hayajneh, A.; Quwaider, M. A Deep Learning Framework for Automatic Detection of Hate Speech Embedded in Arabic Tweets. *Arab. J. Sci. Eng.* **2021**, *46*, 4001–4014. [CrossRef]
45. Anezi, F.Y.A. Arabic Hate Speech Detection Using Deep Recurrent Neural Networks. *Appl. Sci.* **2022**, *12*, 6010. [CrossRef]
46. Mohaouchane, H.; Mourhir, A.; Nikolov, N.S. Detecting Offensive Language on Arabic Social Media Using Deep Learning. In Proceedings of the 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Granada, Spain, 22–25 October 2019. [CrossRef]
47. Alakrot, A.; Murray, L.; Nikolov, N.S. Dataset Construction for the Detection of Anti-Social Behaviour in Online Communication in Arabic. *Procedia Comput. Sci.* **2018**, *142*, 174–181. [CrossRef]
48. Shannaq, F.; Hammo, B.; Faris, H.; Castillo-Valdivieso, P.A. Offensive Language Detection in Arabic Social Networks Using Evolutionary-Based Classifiers Learned from Fine-Tuned Embeddings. *IEEE Access* **2022**, *10*, 75018–75039. [CrossRef]
49. Shannag, F.; Hammo, B.H.; Faris, H. The Design, Construction and Evaluation of Annotated Arabic Cyberbullying Corpus. *Educ. Inf. Technol.* **2022**, *27*, 10977–11023. [CrossRef] [PubMed]
50. Hatebase. Hatebase.org. Available online: https://hatebase.org (accessed on 9 December 2024).
51. Ahmad, A.; Azzeh, M.; Alnagi, E.; Abu Al-Haija, Q.; Halabi, D.; Aref, A.; AbuHour, Y. Hate speech detection in the Arabic language: Corpus design, construction, and evaluation. *Front. Artif. Intell.* **2024**, *7*, 1345445. [CrossRef] [PubMed]
52. Aladeemy, A.A.; Alzahrani, A.; Algarni, M.H.; Alsubari, S.N.; Aldhyani, T.H.; Deshmukh, S.N.; Khalaf, O.I.; Wong, W.-K.; Aqburi, S. Advancements and challenges in Arabic sentiment analysis: A decade of methodologies, applications, and resource development. *Heliyon* **2024**, *10*, e39786. [CrossRef]
53. Mednini, L.; Noubigh, Z.; Turki, M.D. Natural language processing for detecting brand hate speech. *J. Telecommun. Digit. Econ.* **2024**, *12*, 486–509. [CrossRef]
54. Abdelsamie, M.M.; Azab, S.S.; Hefny, H.A. A comprehensive review on Arabic offensive language and hate speech detection on social media: Methods, challenges and solutions. *Soc. Netw. Anal. Min.* **2024**, *14*, 1–49. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Investigations into the Design and Implementation of Reinforcement Learning Using Deep Learning Neural Networks

Roxana-Elena Tudoroiu [1], Mohammed Zaheeruddin [2], Daniel-Ioan Curiac [3], Mihai Sorin Radu [1] and Nicolae Tudoroiu [4,*]

[1] Sciences and Electrical Engineering, University from Petrosani, 332006 Petrosani, Romania; tudelena@mail.com (R.-E.T.); sorinradu@upet.ro (M.S.R.)
[2] Centre for Building Studies, Concordia University from Montreal, Montreal, QC H3G 1M8, Canada; zaheer@encs.concordia.ca
[3] Department of Automation and Applied Informatics, Polytechnic University of Timisoara, 300006 Timisoara, Romania; daniel.curiac@aut.upt.ro
[4] Engineering Technologies Department, John Abbott Collège in Sainte-Anne-de-Bellevue, Montréal, QC H9X 3L9, Canada
* Correspondence: nicolae.tudoroiu@johnabbott.qc.ca; Tel.: +1-514-966-5627

Abstract

This paper investigates the design and MATLAB/Simulink implementation of two intelligent neural reinforcement learning control algorithms based on deep learning neural network structures (RL DLNNs), for a complex Heating Ventilation Air Conditioning (HVAC) centrifugal chiller system (CCS). Our motivation to design such control strategies lies in this system's significant control-related challenges, namely its high dimensionality and strongly nonlinear multi-input multi-output (MIMO) structure, coupled with strong constraints and a substantial impact of measured disturbance on tracking performance. As a beneficial vehicle for "proof of concept", two simplified CCS MIMO models were derived, and an extensive number of simulations were run to demonstrate the effectiveness of both RL DLNN control algorithm implementations compared with two conventional control algorithms. The experiments involving the two investigated data-driven advanced neural control algorithms prove their high potential to adapt to various types of nonlinearities, singularities, dimensions, disruptions, constraints, and uncertainties that inherently characterize real-world processes.

Keywords: centrifugal chiller system; reinforcement learning; deep learning neural network

1. Introduction

Control algorithms are essential parts of the successful operation of processes. In recent years, data-driven intelligent control algorithms mainly relying on artificial intelligence and soft computing concepts have emerged as a valuable alternative to conventional model-based control methods, especially in cases involving complex systems. A special category of such new algorithms that employ the reinforcement learning principle to tune deep neural networks for optimal control of processes is still in the infancy stage but has shown promising results when applied to the control of highly nonlinear, high-dimensional, time-delayed, time-varying, partially known, or hard-to-mathematically-formalize systems. From this perspective, centrifugal chiller systems (CCSs) from complex Heating Ventilation Air Conditioning (HVAC) are no exception, needing specifically tailored control strategies.

In this context, the current paper aims to explore the development and MATLAB/Simulink implementation of two neural reinforcement learning (RL) control algorithms utilizing deep learning neural network (DLNN) frameworks in relation to a specific intricate HVAC centrifugal chiller system. We address the key limitations of existing approaches by making the following contributions:

- We developed an accurate simplified CCS model in a state-space representation. Based on the input–output measurement dataset of the open loop multi-input multi-output (MIMO) CCS extended nonlinear model, having 39 states, three inputs and two outputs, a MIMO autoregressive moving average with an exogenous input (ARMAX) delayed fourth-order polynomial model of z-representation in the complex domain is obtained. This ARMAX model was converted into a linearized MIMO CCS model with four states, three inputs, and two outputs;
- We designed and tuned a standard PID controller and also a standard model predictive control (MPC) for comparison purposes;
- We built two RL DLNN controllers connected in series in a forward path with the MIMO CCS simplified model represented in state space, for temperature control inside the evaporator subsystem and liquid refrigerant level control within the condenser subsystem;
- We rigorously evaluated the tracking performance for both RL DLNNs and compared it with the results obtained using classic PID or MPC controllers.

The remainder of this research paper is structured in the following six sections. Section 2 is devoted to a brief literature review and some preliminaries regarding HVAC control systems. Section 3 outlines four conventional closed-loop control methods, designed and implemented in Matlab/Simulink, with three being based on a standard PID controller, and the fourth being an MPC. Section 4 focuses on the design of two RL DLNN controllers. The extensive simulation results are reported in Section 5. Section 6 is dedicated to discussions, and Section 7 concludes the paper, briefly highlighting the main yields and the future work directions.

2. Related Work and Preliminaries on HVAC Systems

2.1. Literature Review of HVAC Control Systems

It is well known that the electricity market is undergoing substantial transformations in grid modernization, large-scale energy storage, and efficient energy transfer management. Many elements must work together to effectively provide domestic and commercial consumers with the electricity services necessary for sustainable development. Both supply and demand will need to adapt to a new and diverse energy mix, including expanding demand-side management, building new energy storage capacities, and investing in modern and efficient power grids. In this context, a considerable amount of energy consumption in any commercial or residential building is due to heating, ventilation and air conditioning (HVAC) systems [1–4]. Therefore, improving their efficiency becomes critical for energy and environmental sustainability [5–8]. In general, centrifugal chillers are the widely preferred cooling units for a wide range of HVAC control system applications due to their high efficiency, reliability, and low maintenance costs. More precisely, they are suitable for providing cold water for the cooling needs of all the air-handling units in a building [1]. Since a centrifugal chiller is the most energy-consuming HVAC device, its efficiency can be improved by using advanced model-based as well as data-driven controller design strategies [9–11]. A brief review of the literature on HVAC centrifugal chillers reveals that a significant amount of work has been carried out on steady-state and transient modeling. Several dynamic models for the vapor compression cycle have been extensively studied [8]. Also, as previously discussed [1], mechanistic models of single-stage and two-stage cen-

trifugal chillers have been developed in which the centrifugal compressor is modeled based on the Euler turbomachinery, the balance energy, and the impeller velocity equations. The energy coefficient of performance (COP) of the chiller is simulated by considering the compressor polytropic efficiency and hydrodynamic, mechanical, and electrical losses. Meanwhile, the condenser and evaporator are modeled based on the lumped parameter approach, and the heat transfer is calculated based on the effectiveness model. Taking into account the fact that their dynamics are of high nonlinearity, complexity, and dimensionality, from an appropriate control perspective, the developed models must be simplified [8]. An interesting dynamic model of a semi-hermetic reciprocating compressor was developed based on the first law of thermodynamics applied to a lumped control volume, the expansion valve modeled based on a simple orifice flow model [1]. Over the past two decades, our research team has worked intensively with HVAC field control systems, focusing on developing high-fidelity dynamic modeling of centrifugal chillers and building the most suitable control strategies based on them, as a priority task of our research [8]. Interested readers and specialists working in the field can find in [8] a valuable support for the development of simplified, linearized, low-dimensionality, accurate, robust, and stable models in different state-space representations or in the complex domain. Moreover, these models have been validated and adapted to the harshest realistic working environment including a variety of uncertainties, nonlinearities, high dimensionality, and disturbances. Therefore, this paper presents some models of multi-input multi-output (MIMO) centrifugal cooling systems (CCS) with enough details for good readability and understanding. Based on these, advanced closed-loop intelligent neural control strategies of real practical interest were developed. Specifically, the models chosen for our case study served as a valuable vehicle for "proof of concept" and simulation purposes. These models were inspired by the literature in the field, with one of the works being particularly fundamental [12], and others [13–16] providing some valuable and useful references for the design of models based on discrete time data, such as AutoRegressive with eXogenous input (ARX) and Autoregressive Moving Average with eXogenous input (ARMAX), which are polynomial models built in state space. Compared with standard control strategies such as proportional integral derivative (PID) control [17] and also Model Predictive Control (MPC), the advanced neural intelligent reinforcement learning deep learning neural network (RL DLNN) control systems [17–19] demonstrated better performance in various abilities such as possessing human-like expertise in a particular domain, self-adjusting, adaptively learning environmental changes, and taking the best decisions or most appropriate actions [20–24].

2.2. Preliminaries—Adopted CCS Model and Its Implementation

This research used a valuable vehicle, namely three MIMO Centrifugal Chiller System (CCS) models, adopted as a case study, for "proof of concept" and MATLAB R2024a simulation purposes.

2.2.1. MIMO Centrifugal Chiller Modeling Assumptions

The proposed MIMO (three inputs, two outputs) Centrifugal Chiller model used in the case study for this research was previously developed with sufficient detail [8] and also validated [7]. In the current research, this model has been simplified to be suitable for control purposes. In the case study, the proposed dynamic model of the Centrifugal Chiller System was constructed by interconnecting the following subsystems: a water-cooled centrifugal chiller, a centrifugal compressor, shell-and-tube heat exchangers, a thermal expansion valve, and the controller. The water-cooled centrifugal chiller was modeled based on the mechanics of fluid theory, and the centrifugal compressor was modeled based on turbo-machinery theory, similar to those developed previously [8]. The chiller's

capacity control was achieved by the combination of variable inlet guide vane and variable speed drive and the entire system consisted of four major components [1,8], as shown in Figure 1, namely, a centrifugal compressor, a condenser, an expansion valve, and an evaporator. Typically, the chilled-water system consists of one circulating refrigerant loop and two water loops. The first water loop circulates between the condenser and the cooling tower, and the second water loop circulates between the evaporator and the air handling units (AHUs) that produce chilled water for the cooling coil.

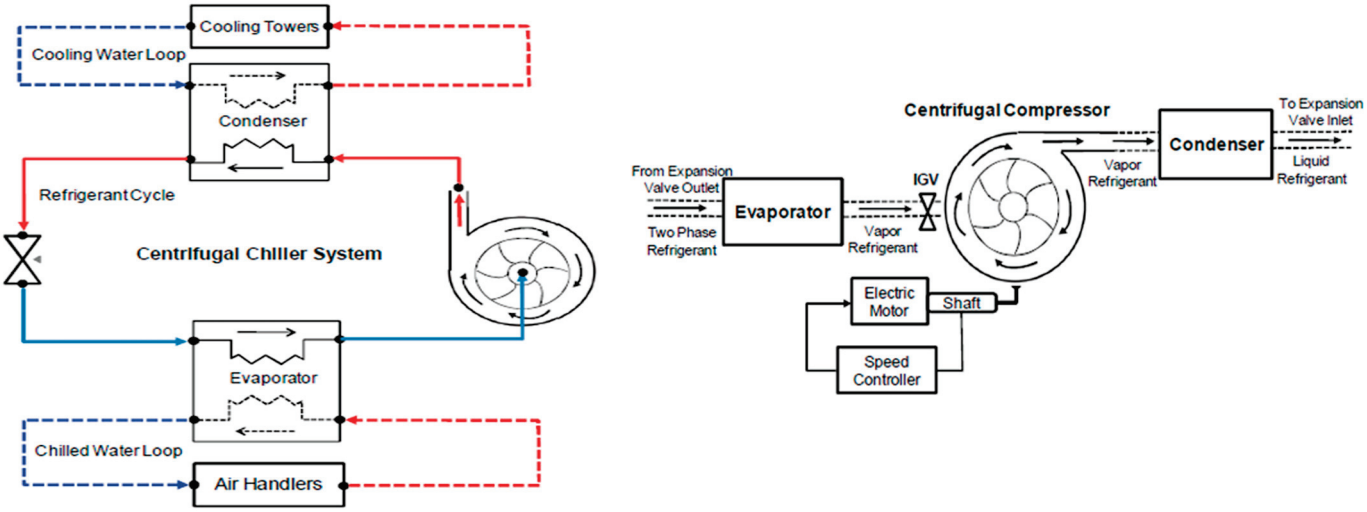


Figure 1. Following the direction of the arrows, chilled water flows through centrifugal chiller evaporator and condenser subsystems, with all four main components [1,8].

A thermal expansion valve is used to regulate the pressure levels at the condenser and evaporator sides. In this research, the overall dynamics of the MIMO chiller control system are calculated in a state-space representation via a set of 39 nonlinear differential equations, with 39 states, two inputs (compressor driver relative speed, expansion valve opening), and two outputs (water temperature, liquid level in the evaporator). The impact of load temperature as a main disturbance was also investigated. Even if a centrifugal chiller is a highly dimensional system, it can easily be integrated into various control HVAC control applications in commercial buildings, with the chilled-water system supplying chilled water for the cooling needs of all the building's air-handling units. In order to accomplish these tasks HVAC, control systems usually include a water pump to circulate the chilled water through the evaporator and throughout the building for cooling. Also, another water loop from the chiller moves the heated condenser water, and another pump circulates the heated water to the cooling tower and cools it back [1,8].

2.2.2. Case Study—MIMO Centrifugal Chiller System Assumptions and Decomposition

The dynamic model of the overall centrifugal chiller control system is of high complexity in terms of state dimensionality and nonlinearity and is described in state space by a set of 39 first-order differential equations (ODE), as a natural mathematical form to represent a physical system. It is beyond the current scope to write out all these equations, since the MIMO Centrifugal Chiller model used in the case study has already been developed and validated in sufficient details in other work [14,15], but we briefly present some significant aspects of the modeling methodology in this section. The first assumption under consideration in the case study related to the decomposition of the overall centrifugal chilller control system into two embedded open-loop subsystems, the first one an open-loop temperature control of the chilled water Tchw_sp inside the evaporator subsystem, and the second an open-loop level control of the liquid refrigerant level L inside the refrigerator subsystem.

The second assumption related to the interference and independence of both loops. Based on two of these assumptions, the three-input two-outputs MIMO model with temperature load disturbance and two independent loops was attached to a data-based CCS model to be investigated. More precisely, the input triplet (compressor speed Ucom, expansion valve opening U_EXV, temperature load disturbance Tchw_rr (chilled water return temperature)) and output doublet (temperature Tchw_sp, level L) were considered. It is worth noting that this modeling strategy, using a deterministic input disturbance Tchw_rr, was the most reflective of real life, which is of particular interest for investigation. As an oriented object, the MIMO Centrifugal Chiller plant is considered in MATLAB as an *iddata* object, generated based on the dataset of input–output measurements loaded in the MATLAB workspace, using the following MATLAB code lines:

load InputOutputChiller_Data.mat *Tchw_sp Level Ucom U_EXV Tchw_rr;*

CentrChiller = iddata (y,u,1)

where the input–output measurement dataset is collected in an open loop using a specific data acquisition equipment and stored in the input vector u = [u1 u2 u3], which is a three-columns vector assigned as follows:

u1 = Ucom; u2 = U_EXV; u3 = Tchw_rr

and in the output vector y = [y1 y2] that is a two-column vector assigned as follows:

y1 = Tchw_sp of the chilled water temperature inside the evaporator subsystem

y2 = Level of the liquid refrigerant level of the refrigerator subsystem

2.2.3. Open-Loop MIMO Centrifugal Chiller System (MIMO) MATLAB Simulink Extended Model Diagram

Representation of the MIMO Centrifugal Chiller System dynamics in state space is achieved with a MIMO Simulink model with three inputs (u1 = Ucom, u2 = U_EXV, u3 = Tchw_rr), two outputs (y1 = Tchw_sp, y2 = Level), and 39 states described by 39 nonlinear first-order differential equations encapsulated in a MATLAB function block, as shown in Figure 2. Also, in Figure 3 are presented the MATLAB Simulink open-loop simulation results.

A detailed Simulink model of an extended MIMO Centrifugal Chiller closed-loop control system (three inputs, two outputs, and 39 states) using a proportional integral derivative (PID) controller described in Section 3, and the MATLAB simulations result is depicted in Section 5.

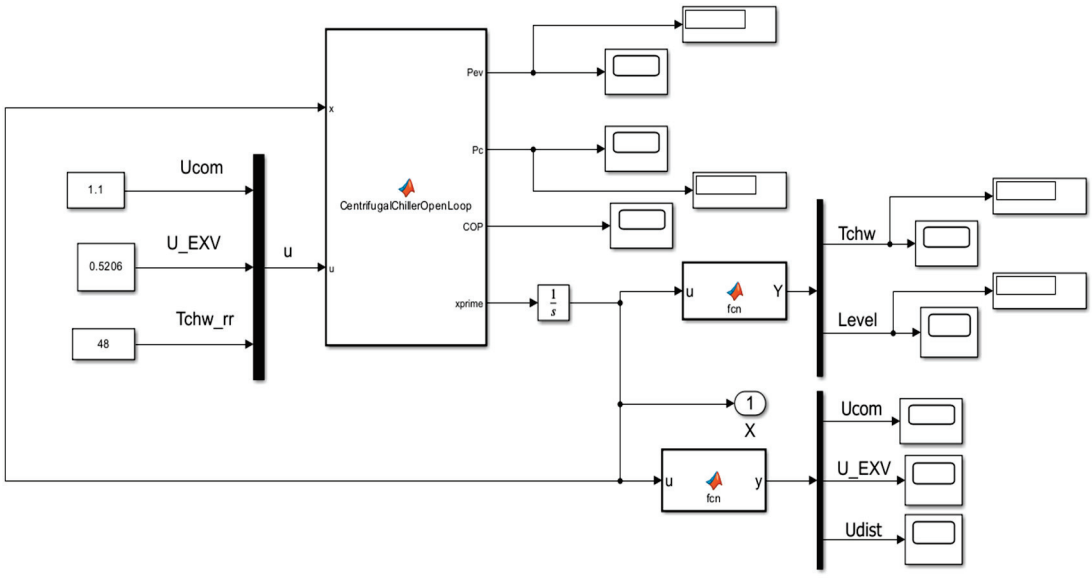


Figure 2. Simulink model of MIMO centrifugal chiller system in state-space representation.

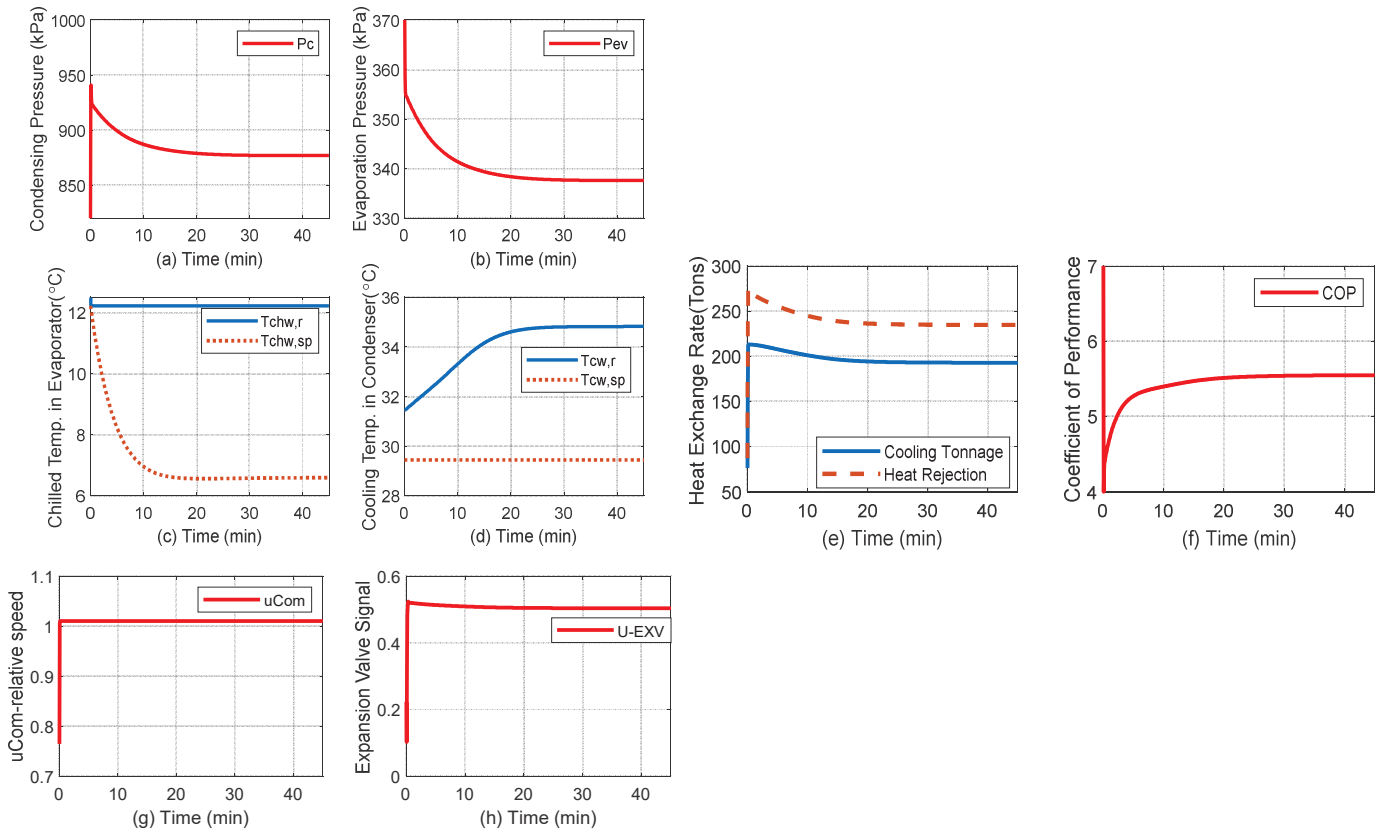


Figure 3. MATLAB Simulink open-loop simulation results: (**a**) condensing pressure (Pc); (**b**) evaporator pressure (Pev); (**c**) chilled water temperature (Tchwr) in evaporator; (**d**) cooling water temperature (Tchwsp) in condenser; (**e**) heat exchange rate; (**f**) coefficient of performance (COP); (**g**) uCom input—relative compressor speed (RPM/RPMnominal); (**h**) u_EXV input–expension valve opening in (%).

2.2.4. Case Study—The Data-Driven ARMAX Model for MIMO Centrifugal Chiller Subsystems in Discrete-Time State-Space Representation

The MATLAB System Identification Toolbox provided valuable tools for developing a simplified second-order polynomial ARMAX model of high accuracy, which was assigned to the full model of the MIMO Centrifugal Chiller. This simplified model proved very useful for building two traditional closed-loop control strategies, namely a PID and MPC, as well as two alternative advanced intelligent reinforcement learning control strategies based on the reinforcement learning deep learning neural networks (RL DLNNs) as a viable alternative to traditional approaches. Finally, a rigorous performance analysis was carried out for comparison purposes, in order to decide which model performed better in terms of meeting the control design requirements and process control constraints, rejecting the effect of possible disturbances acting on the controlled process, accuracy and tracking error. The original MIMO ARMAX model was developed in state-space representation, using a nonlinear set of ordinary differential equations (ODE) as a natural representation of a physical system, easily solved using a suitable MATLAB solver. Due to its nonlinearity and high dimensionality (39 states), a new simplified MIMO Centrifugal Chiller model is developed in a state space representation.

The following two MATLAB code lines generated the MIMO ARMAX (MIMO) discrete state-space model in the z-complex domain [8]:

```
MIMO = armax(CentrifugalChiller_DATA, [[2 0; 0 2],[2 2 2; 3 2 3], [1; 1], [1 1 1; 1 1 1]], options)
```

where the *options* are selected by using the MATLAB code line:

options = armaxOptions('InitialCondition', 'estimate', 'Focus', 'prediction')

In the z-complex domain, the MIMO ARMAX centrifugal chiller model generated in MATLAB is three-input, two-output, four-state linear model described in discrete-time state space by a triplet of matrices (A, B, C) given as follows [8]:

$$A\left(z^{-1}\right) = I_{2\times 2} + \begin{bmatrix} -1.757 & 0 \\ 0 & -1.581 \end{bmatrix} z^{-1} + \begin{bmatrix} 0.7601 & 0 \\ 0 & 0.6884 \end{bmatrix} z^{-2},\ n_a = 2 \tag{1}$$

$$B\left(z^{-1}\right) = \begin{bmatrix} -0.0282 & -0.008134 & 0.001173 \\ 1.469 & -3.07 & 0.03456 \end{bmatrix} z^{-1} + \begin{bmatrix} 0 & 0.001721 & 0 \\ 0 & 2.079 & 0 \end{bmatrix} z^{-2} \tag{2}$$

$$C\left(z^{-1}\right) = I_{2\times 2} + \begin{bmatrix} 0.3064 & 0 \\ 0 & 0.7141 \end{bmatrix} z^{-1} \tag{3}$$

or condensed in discrete-time z-transform transfer matrix representation:

$$H_y^u(z) \stackrel{\text{def}}{=} \begin{bmatrix} H_{y_1}^{u_1}(z) & H_{y_1}^{u_2}(z) & H_{y_1}^{u_3}(z) \\ H_{y_2}^{u_1}(z) & H_{y_2}^{u_2}(z) & H_{y_2}^{u_3}(z) \end{bmatrix} \tag{4}$$

such that:

$$Y(z) = H_y^u(z) \times U(z),\ Y(z) = \begin{bmatrix} Y_1(z) \\ Y_1(z) \end{bmatrix}, \text{ and } U(z) = \begin{bmatrix} U_1(z) \\ U_2(z) \\ U_3(z) \end{bmatrix} \tag{5}$$

where the input and output channels correspond to the following z-transfer functions.

From the channel input u1 to output y1, the discrete-time transfer function is given as follows:

$$H_{y_1}^{u_1}(z) \stackrel{\text{def}}{=} \frac{-0.0282z^{-1}}{1 - 1.756z^{-1} + 0.7601z^{-2}}$$

From the channel input u1 to output y2, the discrete-time transfer function is denoted thus:

$$H_{y_2}^{u_1}(z) \stackrel{\text{def}}{=} \frac{1.469z^{-1}}{1 - 1.628z^{-1} + 0.6884z^{-2}}$$

From the channel input u2 to output y1, the discrete-time transfer function is represented by:

$$H_{y_1}^{u_2}(z) \stackrel{\text{def}}{=} \frac{-0.008134z^{-1} + 0.001721z^{-2}}{1 - 1.756z^{-1} + 0.7601z^{-2}}$$

From the channel input u2 to output y2, the discrete-time transfer function is designated as:

$$H_{y_2}^{u_2}(z) \stackrel{\text{def}}{=} \frac{-3.07z^{-1} + 2.079z^{-2}}{1 - 1.628z^{-1} + 0.6884z^{-2}}$$

From the channel input u3 to output y1, the discrete-time transfer function is given as follows:

$$H_{y_1}^{u_3}(z) \stackrel{\text{def}}{=} \frac{0.001173z^{-1}}{1 - 1.756z^{-1} + 0.7601z^{-2}}$$

From the channel input u3 to output y2, the discrete-time transfer function is defined as:

$$H_{y_2}^{u_3}(z) \stackrel{\text{def}}{=} \frac{0.03456z^{-1}}{1 - 1.628z^{-1} + 0.6884z^{-2}}$$

The Simulink diagram of MIMO centrifugal chiller model in the z-domain including all these z-transfer functions is represented in Figure A1 in Appendix A.

A compact matrix description of the MIMO ARMAX centrifugal chiller system in discrete-time state space (three inputs, four states, two outputs, and sampling time Ts = 1 s) is given by the following state-space equation:

$$x(t+1) = Ax(t) + Bu(t),\ y(t) = Cx(t) + Du(t) \tag{6}$$

where the values of the matrix's quadruplet (A, B, C and D) coefficients are given in Equation (7):

$$A = \begin{bmatrix} 0 & -0.7601 & 0 & 0 \\ 1 & 1.756 & 0 & 0 \\ 0 & 0 & 0 & -0.6884 \\ 0 & 0 & 1 & 1.628 \end{bmatrix},\ B = \begin{bmatrix} 0 & 0.006884 & 0 \\ -0.1128 & -0.03254 & 0.004691 \\ 0 & 1.04 & 0 \\ 0.7347 & -1.535 & 0.01728 \end{bmatrix},\ C = \begin{bmatrix} 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix},\ D = 0_{2\times 3} \tag{7}$$

The MIMO output responses with the disturbance temperature set to Tchw-rr = 48 [°F] and the actuator inputs (Speedcomp, ExpValve) are shown in Figure 4.

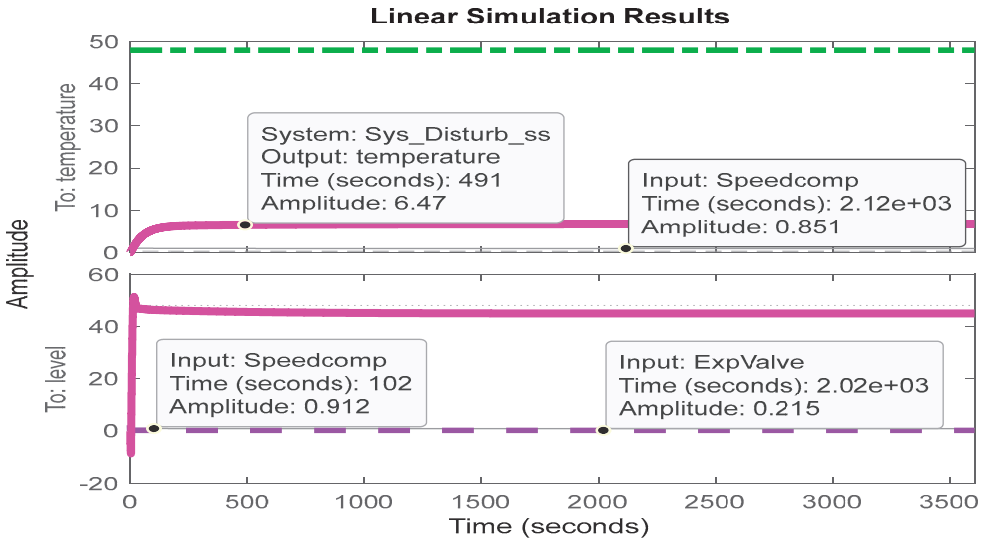


Figure 4. Open-loop linear simulation results for the MIMO ARMAX CCS state-space model.

3. Traditional Controllers: PID and MPC Closed-Loop Control Strategies

In the time domain, the general form of the PID control law in a closed-loop control strategy is given by the following input–output equation:

$$u_{PID}(t) = k_P\varepsilon(t) + k_I \int \varepsilon(t)dt + k_D \frac{d\varepsilon(t)}{dt} \tag{8}$$

where $\varepsilon(t)$ represents the error between the actual measurement of system output $y(t)$ and its desired value $ysp(t)$, also called the setpoint, reference, or tracking value, as follows:

$$\varepsilon(t) = y(t) - ysp(t) \tag{9}$$

where $u_{PID}(t)$ refers to the controller output.

$k_P, k_I = \frac{1}{T_I}$, $k_D = T_D$ represent the PID controller parameters, k_P is the proportional gain, k_I refers to the integral gain, k_D denotes the derivative gain, and T_I and T_D are the time constants for the integral and derivative blocks, respectively. For a particular parameter setting, different controller architectures (e.g., P, I, PI, PD, and PID) may be obtained.

In the complex domain, the transfer function of an ideal PID controller is given bellow:

$$H(s) = \frac{U_{PID}(s)}{\varepsilon(s)} = k_P + \frac{k_I}{s} + k_D s \tag{10}$$

Furthermore, adding a compensator formula to Equation (10) using a low pass filter (LPF) of coefficient N can serve as a practical PID controller via the following transfer function:

$$H(s) = \frac{U_{PID}(s)}{\varepsilon(s)} = k_P + \frac{k_I}{s} + \frac{k_D N}{1 + \frac{N}{s}} \tag{11}$$

3.1. DTI MIMO Centrifugal Chiller Closed-Loop System Control in State-Space Representation

The discrete-time integrator (DTI) closed-loop control strategy architecture of the proposed MIMO CCS model, a valuable "vehicle" for MATLAB Simulink simulations, is depicted in Figure A2, Appendix A. Its discrete-time transfer function in the z-complex domain is given as follows:

$$H_1(z) = \frac{k_1 T_s}{1 - z^{-1}},\ k_1 = [-1\ 1],\ H_1(z) = \frac{k_2 T_s}{1 - z^{-1}},\ k_2 = [0.001 - 0.01], T_s = 1 \tag{12}$$

The parameters of both DTI controller blocks k_1, k_2 are easy to adjust. However, the parameter adjustment process encounters some difficulties due to the constraints on the model inputs $0 \le u_1 =$ uCom ≤ 1.1, $0 \le u_2 =$ u_EXV ≤ 1. Also, the "trial and error" procedure to track evaporator temperature and condenser level performance was inaccurate. Figure A2a presents the compact DTI Simulink diagram, with valuable details of the two integrators shown in Figure A2b,c.

3.2. PID MIMO Centrifugal Chiller Closed Loop System Control in Extended State Space Representation

The PID closed-loop control strategy for the MIMO CCS extended model with 39 internal states is depicted in Figure A3a–c. The PID transfer function in the s-complex domain is given by Equation (11). Both controllers' parameter datasets for evaporator temperature (indexed by T) and condenser liquid refrigerant level (indexed by L) were set to the following values:

$$k_{PT} = 0.1,\ k_{IT} = 0.0001,\ k_{DT} = 0.1,\ N_T = 100,\ k_{PL} = 0.1,\ k_{IL} = 0.0001,\ k_{DL} = 0.1,\ \ N_L = 100,$$

The temperature setpoint within the evaporator was Tchw-sp = 6.67 [degC], and the level of liquid refrigerant inside the condenser was L = 45 (%), with a much longer simulation time of Tf = 13,600 s (3 h 30 min).

Therefore, alternative simplified ARMAX, ANFIS and neural network models of the MIMO Centrifugal Chillaer capable of capturing its entire dynamic evolution and suitable for control purposes were investigated in this research.

Figure A3a shows the Simulink diagram of the PID closed-loop control strategy, and Figure A3b presents two subsystem components of the Simulink diagram (model + visualization block). The code lines of the Simulink MATLAB function from the visualization block are given in Figure A3c. The Simulink simulation results are discussed in Section 5, and a rigorous performance analysis is conducted in Section 6.

3.3. Digital PID Control of MIMO Centrifugal Chiller Closed-Loop System in Extended State-Space Representation (39 States)

A large number of simulations performed on a MATLAB Simulink R2024a platform showed a slow response of about 3 h and 30 min to reach a steady state and achieve

accurate tracking performance for control of the evaporator subsystem temperature and the levels of liquid refrigerant in the condenser subsystem. The development of a fast-tuning digital PID controller, similar to an interesting approach proposed previously [15], provided a substantial improvement. Furthermore, an alternative to the extended MIMO centrifugal chiller model, a simplified Adaptive Neural Fuzzy Inference System (ANFIS) is under investigation for integration with the digital PID in the same closed-loop control strategy whose Simulink block diagram is shown in Figure 5. The MATLAB Simulink simulations are presented and discussed in Section 5 and a rigorous performance analysis is carried out in Section 6, revealing a fast response (settling time) and high tracking accuracy. Furthermore, the model's performance was compared with an advanced deep learning neural network (RL DLNN) in which the reward function was generated based on the step response block specifications of digital PID control of the MIMO CCS ANFIS model developed in the next section.

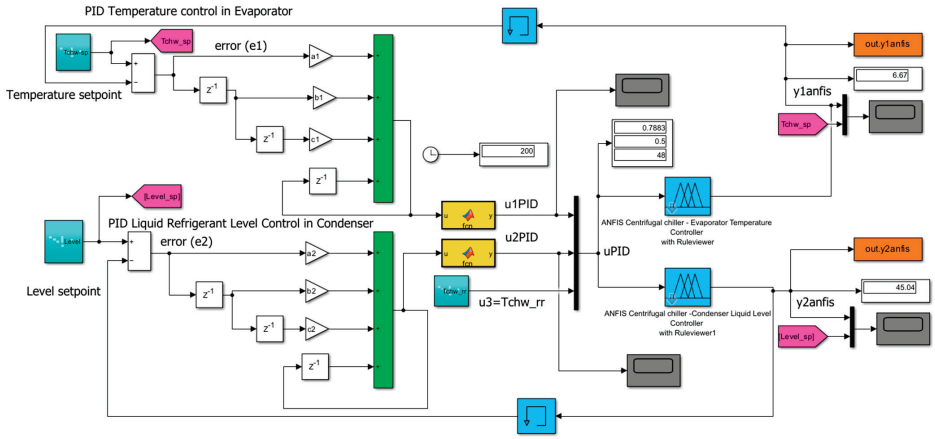


Figure 5. Digital PID ANFIS discrete-time control temperature inside evaporator and liquid refrigerant level within condenser—overall Simulink diagram.

In the Simulink diagram shown in Figure 5 the digital PID controller appears on the left side, implemented based on Equation (8). The simplest way to implement a digital PID is to discretize Equation (8) and then to transform it into a difference equation of the form suggested in [22]. Specifically, this is a recursive method that calculates the PID controller output at $t = kT_s,\ k \in \mathbb{Z}$, T_s—sampling time, based on the previous value of the controller output and its growth [22], as follows:

$$u_{PID}(kT_s) = u_{PID}[(k-1)T_s] + \Delta u_{PID}(kT_s) \tag{13}$$

For simplicity, assuming $T_s = 1[s]$, Equation (13) becomes:

$$u_{PID}(k) = u_{PID}(k-1) + k_P(e(k) - e(k-1)) + k_I e(k) + k_D(e(k) - 2e(k-1) + e(k-2)) \tag{14}$$

or, compact:

$$u_{PID}(k) = u_{PID}(k-1) + (k_P + k_I + k_D)e(k) - (k_P + 2k_D)e(k-1) + k_D e(k-2) \tag{15}$$

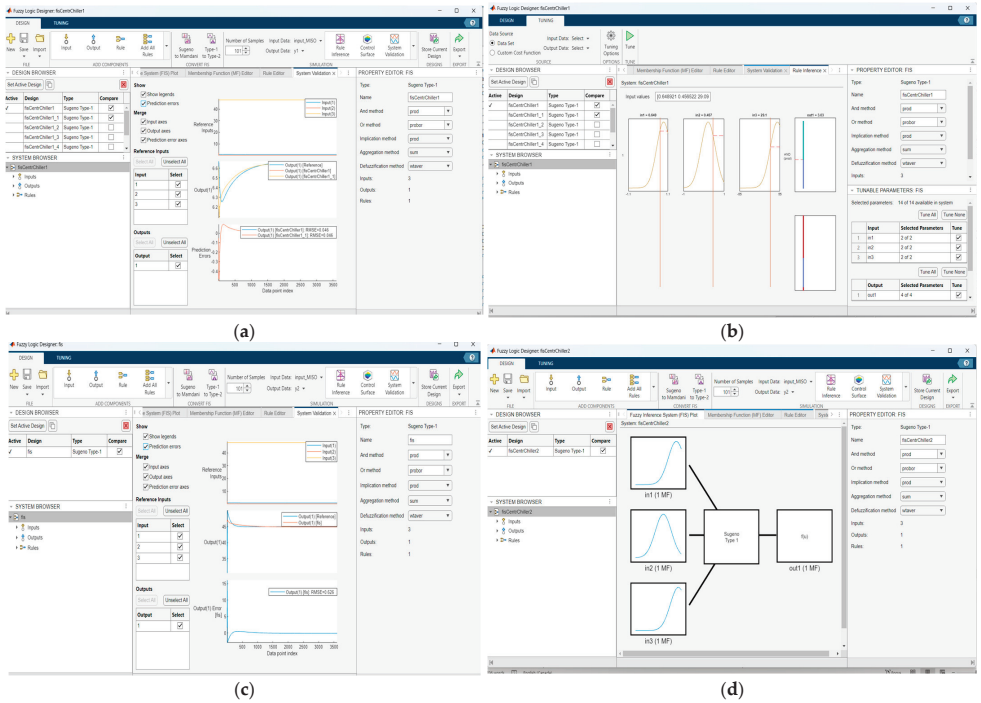
$$u_{PID}(k) = u_{PID}(k-1) + ae(k) + be(k-1) + ce(k-2) \tag{16}$$

with the coefficients a, b and c given by:

$$a = k_P + k_I + k_D,\ b = -k_P - 2k_D,\ c = k_D \tag{17}$$

In this research, the controller coefficients were set to (0.095, −0.225, 0.1) for the first digital PID controller integrated into the closed-loop feedback control of the chilled water temperature inside the ANFIS model evaporator subsystem.

The set of values (0.025, −0.105, 0.05) was selected for the second digital PID controller that controlled the liquid refrigerant level in the ANFIS model condenser subsystem. As can be seen, the MIMO Centrifugal model was split into two accurate ANFIS models of the MIMO centrifugal chiller; the first a MISO ANFIS CentrChillerfis1 attached to the evaporator subsystem, and the second a MISO ANFIS CentrChillerfis2 assigned to the condenser, as shown in Figure 6a–h. Both MISO ANFIS models were generated using the Fuzzy Logic Designer app based on the measurement input–output dataset of the open-loop MIMO Centrifugal Chiller plant. The first ANFIS MISO evaporator centrifugal chiller model (fisCentrChiller1) is shown along with the membership functions in Figure 6a,b, and its estimation performance is reported in Figure 6e. Data for the second ANFIS MISO condenser centrifugal chiller, including the membership functions, are shown in Figure 6c,d and its estimation performance in Figure 6f. Figure 6g,h show a Rule Viewer for each MISO ANFIS model.



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

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

Figure 6. *Cont.*

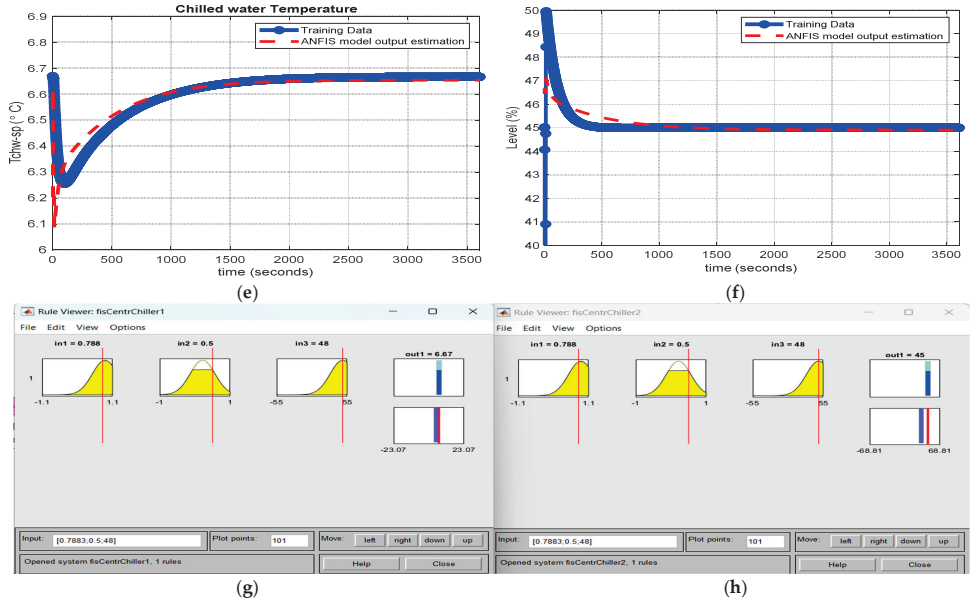


Figure 6. Fuzzy logic design application, MATLAB Simulink simulation results: (**a**) Centrifugal Chiller FIS temperature within evaporator; (**b**) Centrifugal Chiller FIS membership functions; (**c**) Centrifugal Chiller FIS temperature within condenser; (**d**) Centrifugal Chiller FIS membership functions; (**e**) Chilled water temperature, evaporator training data and ANFIS model output estimation; (**f**) liquid refrigerant level inside condenser training data and ANFIS model output estimation; (**g**) Rule Viewer: fis MISO chilled water temperature inside the evaporator subsystem; (**h**) Rule Viewer: fis MISO liquid refrigerant level within the condenser subsystem.

The MATLAB Simulink simulation results are discussed in Section 5, and a rigorous comparative performance analysis is made in Section 6.

3.4. Model Predictive Control Based on Centrifugal Chiller MIMO State-Space Representation

The MPC closed-loop control strategy architecture for MIMO CCS is shown in Figure 7. The model predictive control (MPC) object was generated in MATLAB, as explained is detail in Section 4.2, and the MPC Simulink diagram shown in Figure 7 was obtained using the MPC MATLAB Simulink Toolbox. The MPC object has two manipulated variables (MVs): uCom and u_EXV; one measured disturbance (MD) (temperature Trr set to 48 degrees Fahrenheit); and two measured outputs variables (OV), namely, evaporator temperature Tchw_sp and liquid refrigerant level inside condenser. The MIMO model of the Centrifugal Chiller plant is represented in a discrete-time state space via Equation (6), with the values of the matrices' coefficients given in Equation (7). The MATLAB Simulink simulation results are shown in Section 5.2 and the performance analysis is reported in Section 6.

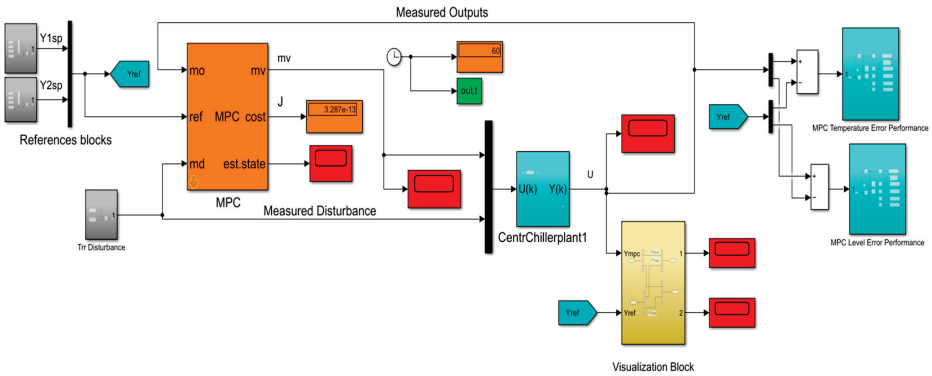


Figure 7. Simulink models of MPC using Simulink MPC Toolbox and MPC Design application.

4. Design and Implementation of Reinforcement Learning Using Deep Learning Neural Networks-MIMO CCS Closed-Loop Control Strategies

This section focuses on the design and implementation in MATLAB Simulink of two RL DLNN advanced intelligent control strategies for a MIMO CCS simplified state-space model. The first uses the MPC specifications to generate the reward function, and the second uses step response specifications imposed on a MIMO CCS model represented in the simplified state space. The tracking accuracy performance was compared with the MPC integrated in the first controller structure and an improved digital PID based on the MISO CCS ANFIS models.

4.1. Reinforcement Learning Closed-Loop Control Strategy Using Deep Learning Neural Network MIMO Centrifugal Chiller Plant Model Represented in State Space

4.1.1. Reinforcement Learning Process—Description

Reinforcement learning (RL) involving multi-layered deep learning neural networks is a well-used and well-suited method for research and development in modern artificial intelligence, as is shown in [18]. More precisely, RL is a modeling process in which an agent (controller) learns to make decisions by interacting with an unknown environment through trial and error, as shown in Figure 8a [18,19]. Mathematically speaking, the RL modeling process (algorithm) depicted in Figure 8a is typically based on a Markov decision process (MDP). In this type of process, the RL agent block receives in its current state the observation St as a result of the interaction with the unknown Environment, performing an action that it then sends to the same Environment. The latter reacts by sending a scalar reward Rt to the same RL block for a new transition to the next state St+1, according to the conditional probability of the Environment dynamics; p(St+1 | St, At). The RL agent attempts to learn a policy π(At | St) or map from observations to actions, in order to maximize its returns (expected sum of rewards). In RL (as opposed to optimal control), the algorithm has access to Environment dynamics p(St+1 | St, At) only during the sampling time. During the training process, the RL blocks learn the unknown environment and "make a series of decisions to maximize the cumulative reward for the task without human intervention and without being explicitly programmed to achieve the task" [18].

From the perspective of a system-oriented approach, the diagram shown in Figure 8a shows a general representation of a reinforcement learning (RL) algorithm. Figure 8b reveals its equivalence to a traditional closed-loop control strategy used in system applications [19]. In other words, the key role of the RL policy in Figure 8a is to observe the unknown environment (state St) and generate actions (At) to "complete a task optimally", similar to a traditional controller operating in a control system application. Of course, the RL algorithm comes with a valuable improvement to the RL policy, known as policy updating. More specifically, RL can be seen as a mapping of a conventional feedback control system depicted in Figure 8b, and the correspondence between both diagrams is well presented in [19]. The concept is based on "*rewarding or punishing an agent's performance in a specific environment*" [24]. According to this definition, a state is a description of the environment that provides the agent with helpful information for taking a decision at each time step. The agent receives observations and a reward from the Environment and the actions generated are sent into the same Environment. Reward is a measure of the achievement of an action on the completion of the task goal; i.e., the reward signal evaluates the outcome of past actions [18]. RL policy is based on cost function to map each state to the optimal action in order to maximize its reward function during the episode [17].

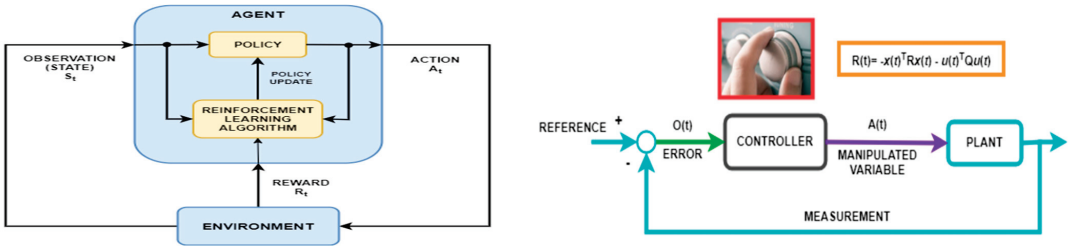


Figure 8. Reinforcement learning architecture [18,19]: (**a**) RL process; (**b**)standard feedback closed-loop control schematic. In this figure, the blue color is used to identify the Agent and Environment blocks, the color green to show the direction of the information flow in the closed-loop of traditional system, and the red color is to highlight that the parameters of the controller are adjustable and a particular Reward function, respectively.

This RL policy can be a deterministic policy, defined as $A_t = \pi(S_t)$ for $\pi : \mathcal{S} \to \mathcal{A}$, or stochastic policy, described as $A_t \sim \pi(\cdot|S_t)$ for $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\mathcal{A})$ is a set of probability distributions over actions. For an MDP, these probability distributions have an interesting feature; they factorize over trajectories [17,19]:

$$p(S_1, A_1, \ldots, S_n, A_n) = p(S_1)\pi(A_1, S_1)\,\mathcal{P}(S_2|S_1, A_1)\pi(A_2, S_2)\,\ldots\ldots\mathcal{P}(S_n|S_{n-1}, A_{n-1})\pi(A_n, S_n) \tag{18}$$

Thus, this feature is useful in the field of control applications, since RL policies need only to consider the current state, which is a strong consequence of the Markov assumption and full observability [23]. Otherwise, if the RL environment is partially observable, then the RL policy must depend on the history of observations. It is important to note that the RL deep learning method differs from supervised learning because it does not require correct sets of actions and labelled input/output pairs [24–27]. The general reinforcement learning structure shown in Figure 8a is described by the following Equations [17]:

$$A_t \sim \pi(A_t|S_t) \tag{19}$$

$$S_{t+1} \sim f_{state}(S_{t+1}|S_t, A_t) \tag{20}$$

$$R_{t+1} = f_{reward}(S_t, A_t, S_{t+1}) \tag{21}$$

$$R = \sum_{t=0}^{t=\infty} \gamma^t R_{t+1} \tag{22}$$

$$Q^{new}(S_t, A_t) = Q^{old}(S_t, A_t) + \alpha\left(R_t + \max_{a' \in \mathcal{A}} Q^{next}(S_{t+1}, a') - Q(S_t, A_t)\right) = Q^{old}(S_t, A_t) + \alpha(Target - Prediction) \tag{23}$$

where $\pi(A_t|S_t)$ is the policy at time t, A_t denotes the action of time t, S_t is the state (observation) at time t, f_{state} and f_{reward} are the transition functions from time t to t+1, and Q is the state-action value function. The so-called Q function, $0 \leq \gamma < 1$ is an important hyperparameter called the discount factor, which determines how much we care about rewards now versus rewards later; more precisely, this is an exponential decay factor, which means longer term planning is harder.

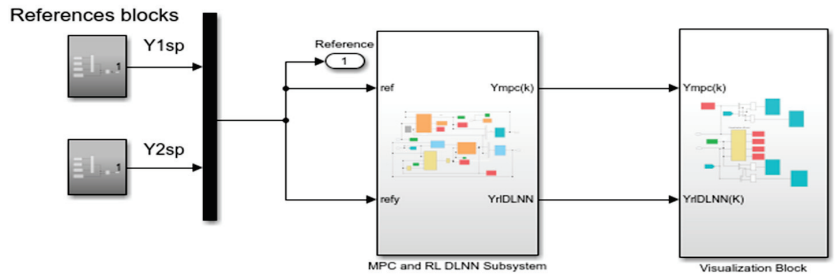
Also, $0 < \alpha < 1$ represents the learning rate—a value that controls how quickly an agent updates its Q-values based on new information, essentially determining how much weight is given to newly acquired experiences compared with previously learned information; a higher value for α means faster learning and more significant updates to Q-values with each new interaction. When adopting a Monte Carlo technique to estimate the Q-values, every iteration of the Q-value requires updating Q for every state. Monte

Carlo estimation of the expectation $\mu = \mathbb{E}[X]$ repeatedly samples X, and the convex relationship [23] is updated as follows:

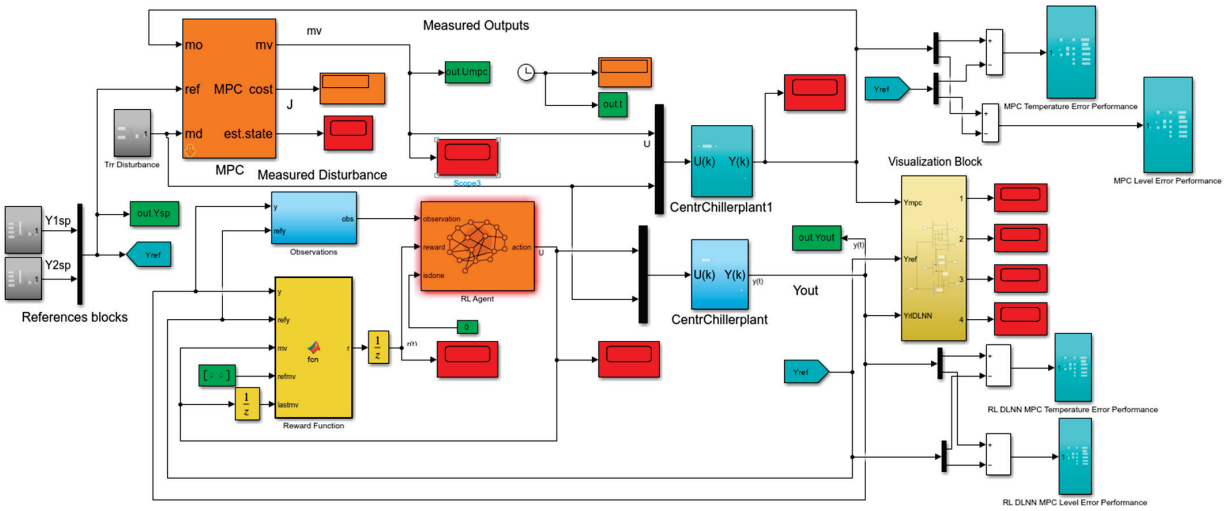
$$\mu \leftarrow \mu + \alpha(X - \mu) \tag{24}$$

In this relationship, the X dynamics is required to evaluate the expectation μ.

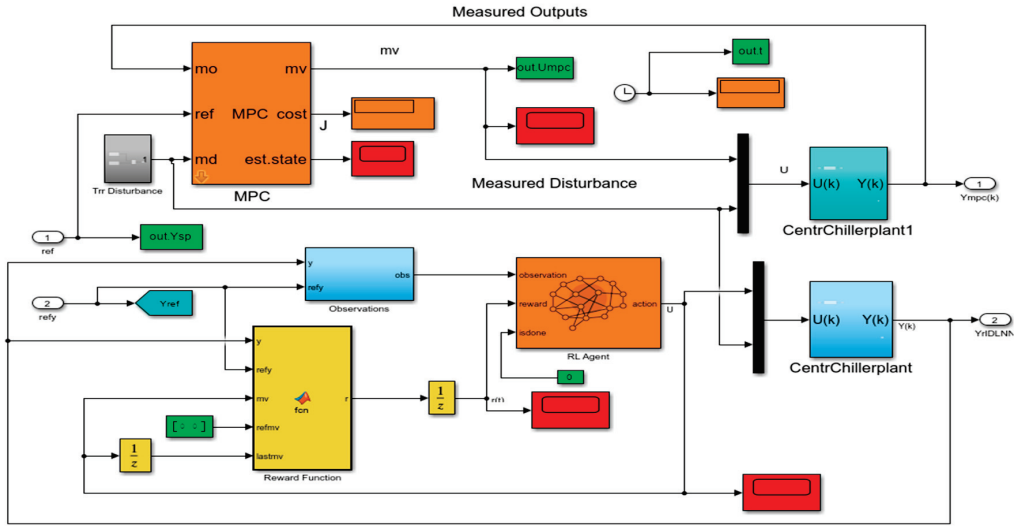
The ε-greedy policy that appears in Figure 9f for a block of parameters of the TD3 RL agent is defined as a policy that chooses $argmax_{a \in \mathcal{A}} Q(S, A)$ with probability 1-ε and a random action with probability ε. The typical value for the parameter ε is 0.005 [23].



(**a**)

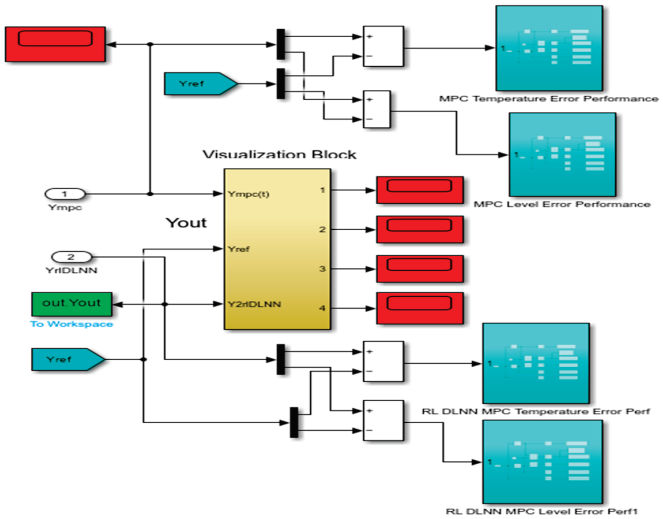


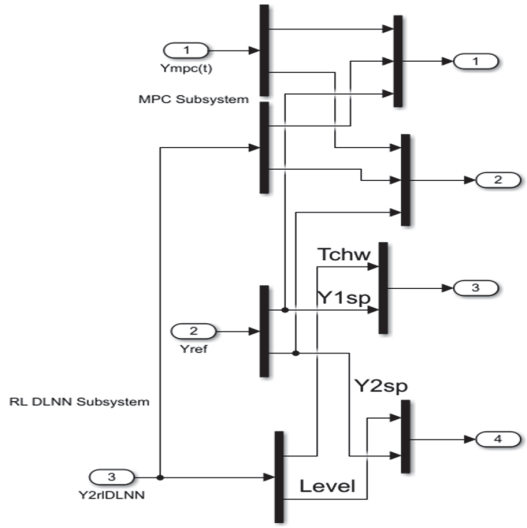
(**b**)



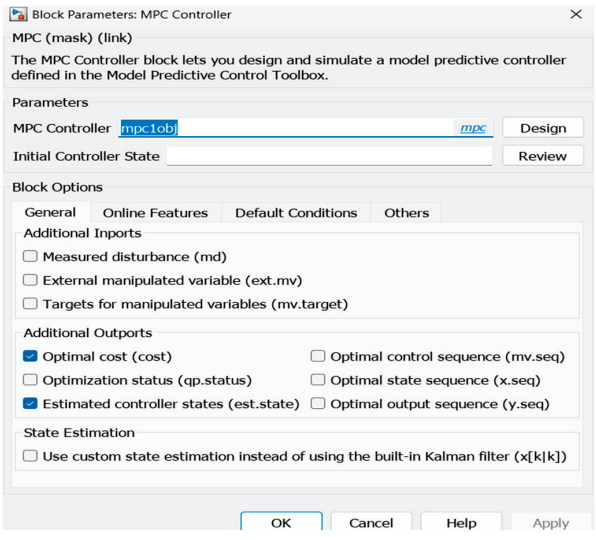
(**c**)

Figure 9. *Cont.*

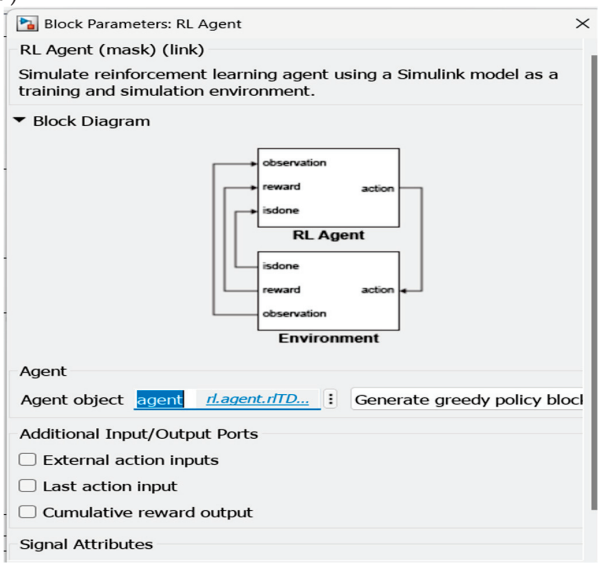




(**d**)



(**e**)



(**f**)

Figure 9. *Cont.*

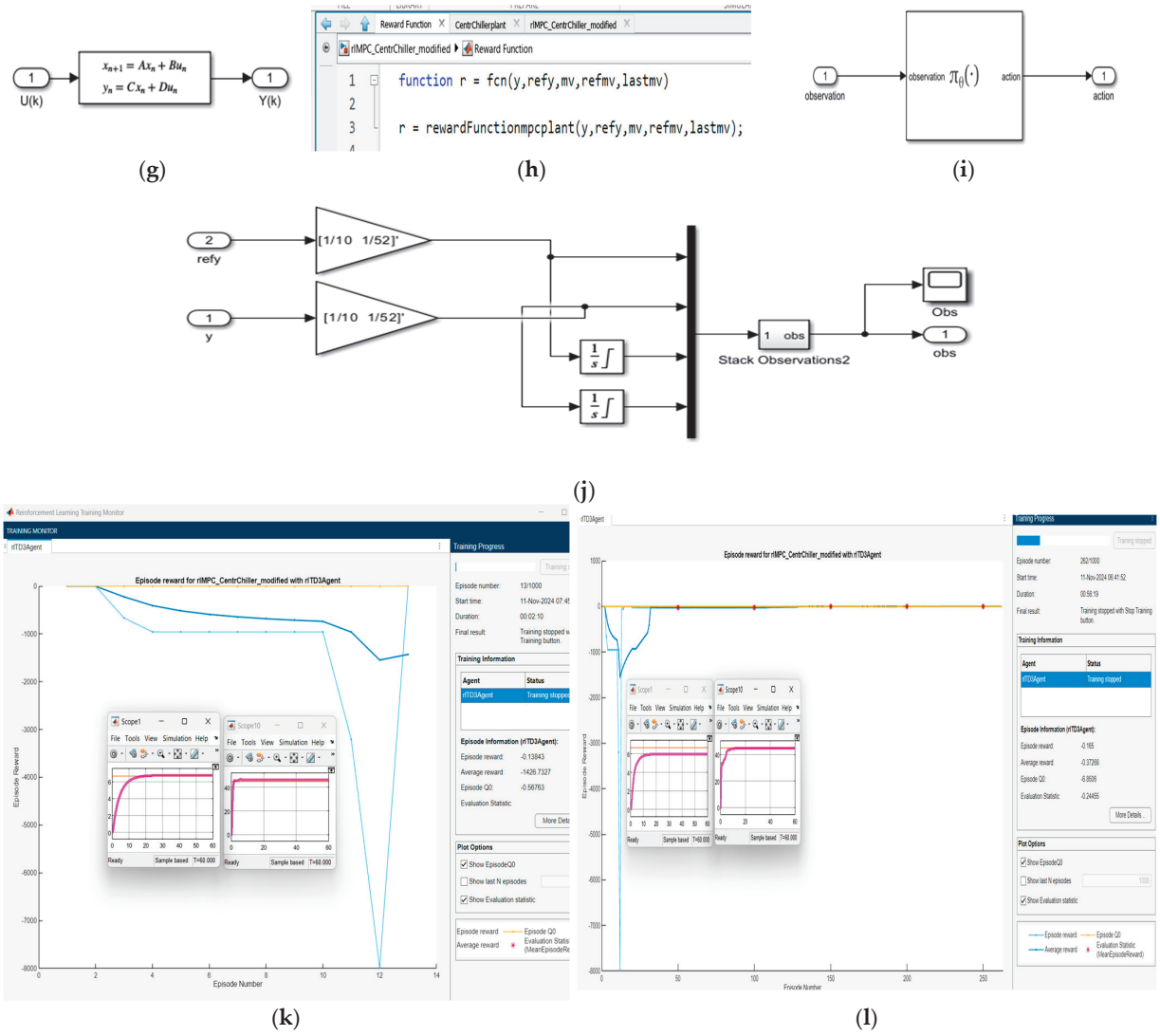


Figure 9. Reinforcement learning control strategy based on deep neural network for a MIMO Centrifugal Chiller simplified model in state-space representation: (**a**) compact Simulink diagram of MIMO Centrifugal Chiller plant and RL DLNN; (**b**) detailed Simulink diagram; (**c**) MPC and RLDNN subsystem; (**d**) visualization block; (**e**) MPC parameters block; (**f**) RL Agent block (controller); (**g**) state-space representation block; (**h**) reward function; (**i**) RL Agent greedy policy block; (**j**) observation block-detail; (**k**) agent training process-print screen snapshot after 13 epochs; (**l**) agent training process-print screen snapshot after 262 epochs.

Combining all the information on Monte Carlo estimation of the expectation μ, a helpful pseudocode algorithm for implementers called Q-learning with ε-greedy policy is given in [23].

The ε-greedy policy is a valuable mechanism for managing the exploration–exploitation balance, as has been stated [23]. It operates according to the following equation:

$$\pi_{\varepsilon}(\mathcal{S};Q) = \begin{cases} argmax_{a\in\mathcal{A}}Q(S_t,\ \mathrm{A}) \text{ with probability } 1-\varepsilon \\ Uniformly\ random\ action\ in\ A\ with\ probability\ \varepsilon \end{cases} \tag{25}$$

Due to its simplicity, the ε-greedy policy has become the most widely used method to balance exploitation and exploration. It ensures that most of the time (probability 1-ε), the RL agent exploits its incomplete knowledge of its environment by choosing the best action, and occasionally (probability ε) explores other actions. There may be some good actions that the agent will never find without exploration [23]. This is why the

exploration–exploitation trade-off is an important research topic. Also, for our readers and implementers interested in being well-documented, the pseudocode Algorithm 1 of the RL Deep Learning NN algorithm described in [17] is reproduced below for its great informal value:

Algorithm 1: RL Deep Learning NN [17]

RL Agent states: Observations $\mathcal{S}$
RL Agent actions; $\mathcal{A}$
Define the optimal policy given by the Bellman equation [23]:

$$Q^{\pi^*}(s,a) = r(s,a) + \gamma(\sum_{s'} p(s'|s,a)\max_{a'} Q^{\pi^*}(s',a') \tag{26}$$

and
$Q^* = Q^{\pi^*}$ is the optimal state-action value function;
Initialize the hyperparameter learning rate α, discount factor γ, and exploration parameter ε.
for t in range (epoch) do
Calculate the action A_t according to the *optimal policy*:

$$A_t \leftarrow \begin{cases} argmax_{a\in\mathcal{A}} Q(S_t,\ \mathrm{A}) \text{ with probability } 1-\varepsilon \\ Uniformly\ random\ action\ in\ A\ with\ probability\ \varepsilon \end{cases} \tag{27}$$

Send the action A_t into the agent environment;
Evaluate the next value of the observation S_{t+1} starting from the previous state S_t according Equation (20) and [23]:

$$S_{t+1} \sim f_{state}(S_{t+1}|S_t, A_t) = \mathcal{P}((\ \cdot|S_t,\ A_t)) \tag{28}$$

The agent is sensed by the next observation S_{t+1} and the reward function R_{t+1} given by Equations (21) and (22) (could be r(S_t, A_t), or could be stochastic);
Update the new action-value function $Q^{new}(S_t, A_t)$ at state-action (S_t, A_t), according the Equation (23);
Evaluate the estimation using the following loss function in terms of RMSE, MSE, and MAE as defined in [17].
End

The MATLAB software package Simulink Reinforcement Learning Toolbox provides all the valuable tools to create and train reinforcement learning agents (controllers), as previously demonstrated [20,21]. In many practical decision-making problems, the MDP states are high-dimensional and cannot be solved by traditional RL algorithms. Deep reinforcement learning algorithms incorporate deep learning (DL) to solve such MDPs, often representing the $\pi(A_t \mid S_t)$ policy or other learned functions as a neural network (NN) and developing specialized algorithms that perform well in this context. Therefore, in this study, the agent policy (control law) was implemented using deep neural networks (DNNs) created using the most appropriate tools provided in the MATLAB Simulink Deep Learning Toolbox software package [24–26]. Some problems related to the graphics issues encountered during the MATLAB simulations were solved based on the advice provided in [27].

4.1.2. Reinforcement Learning Workflow

The following steps are suggested in [19] to illustrate the RL workflow:

Step 1. Formulate problem: define the task for the RL agent to learn, including its interaction with the environment and primary and secondary achievement goals;

Step 2. Create environment: employ MATLAB or Simulink to define the environment within which the RL agent will operate, including the interface between the agent and environment and the environment dynamic model;

Step 3. Create RL Agent: create the RL agent containing *policy* and a *learning algorithm* [18,19];

A learning algorithm continuously updates the policy parameters based on the actions, observations, and rewards;

Step 4. Create Deep Neural Network Policies and Value Functions: as discussed in Section 4.1.1.

The agents define the DLNN inputs and outputs based on specified actions and observations from the Environment. Also, an actor net and an actor critic representations for the RL agents can be created using Deep Learning Toolbox functionality. Pretrained deep neural networks or deep neural network layer architectures can be imported using the MATLAB Simulink Deep Learning Toolbox network import functionality.

Step 5. Train RL Agent: the RL Agent policy representation is trained using the defined environment, reward, and agent learning algorithm. Training an agent using reinforcement learning is an iterative process.

Step 5. Validate RL Agent: Evaluate the performance of the trained agent by simulating the agent and environment together.

Step 6. Deploy policy: Deploy the trained policy representation using a generation code.

4.2. Deep Learning Neural Network

In this section, the deep learning neural networks (DLNN) is discussed. A deep learning neural network is structured as a series of interconnected layers with multiple nodes called "neurons". Deep learning is used for training multi-layered, allowing them to learn hierarchical representations of datasets in order to make predictions and to generate outputs.

4.2.1. DLNN Architectures, Components, Algorithms and Applications

The deep learning (DL) architecture originated from artificial neural networks (ANNs) that were inspired by the structure of the human brain. Neural network (NN) architectures consist of several components working together to process and learn from massive datasets, extracting patterns and abstract representations of data as information flows through each layer. The components include the input layer, hidden layers, neurons (nodes), weights and biases, activation functions, the output layer, and the loss function; these are well explained in a comprehensive review presented in [24]. In that review, deep learning is described as a multi-layer machine learning method for performing complex tasks. This layered structure is why the deep learning technique differs from conventional machine learning methods; within this structure, the data flows from an input layer through several hidden layers to an output layer. These added hidden layers increase the accuracy performance of the neural network design.

Each neuron performs calculations on the received input and passes the result to the next layer, allowing the network to learn complex patterns from the data [24]. Also, a perceptron simulates a neuron with a set of inputs, each of these having a particular weight, such that the inputs with higher weights have a significant impact on the neural network. The neuron computes and produces an output based on the weighted inputs. Each neuron receives n inputs (features), sums them, applies a transformation (activation) and generates the output. A second parameter called bias is used to adjust the output based on the input weights; thus the model fits the data in the best possible way. An activation function converts the inputs into the output produced using a threshold. Briefly, the neural network architecture serves as the support for the understanding and processing of diverse data types, and generative models unlock the ability to create new data samples that resemble the training data. In the literature [24], it is mentioned that deep learning is "among the fastest-growing areas in computational science, employing complex multilayered networks to model high level patterns in data". The state-of-the-art of deep learning techniques has

been enhanced in several fields, including "visual object recognition, speech recognition, genomics, and discovery of drugs, along with plenty of others". Among the well-known deep learning algorithms discussed [24] are backpropagation models, autoencoders, variational autoencoders, restricted Boltzmann machines, deep belief networks, convolutional natural networks, forward neural networks, recurrent neural networks, LSTM, generative adversarial networks, transformer deep learning structure, embedding from language models, and bidirectional encoder representations from transformers. A brief overview of the steps involved in using deep learning is expressed in the following sequence:

$$1.\textit{Data Preparation} \rightarrow 2.\textit{Building NN} \rightarrow 3.\textit{Training NN} \rightarrow 4\frac{\textit{Evaluation}}{\textit{Tuning}} \rightarrow 5\frac{\textit{Prediction}}{\textit{Generation}} \rightarrow 6.\textit{Iterative Improvement}$$

These steps can be seen in the MATLAB code lines of both the RL DLNN algorithms developed in the present research, Appendix B.1, Algorithm A1, and Algorithm A2 respectively.

In the first step, the data are collected and preprocessed (cleaning by removing errors or inconsistences, normalization, and splitting, such as for example, 70% into training data and 30% testing data). The data format must be compatible with the NN. The architecture of the NN is defined in the second step. In the third step, the training process is launched for fine-tuning of the network's weights and biases in order to minimize the difference between the predicted output and the true output. This is followed by step 4, evaluation and validation, required for a rigorous evaluation of performance accuracy, followed by a test validation to gather valuable information about how well the NN generalizes to unseen data. In step 5, the trained DLNN evaluated and validated in the previous step acquires good ability to predict or generate results for unseen data. If the proposed DLNN performance does not meet the control objective in the last step, an iterative improvement process is initiated to adjust the DLNN parameters, including modifications to the DLNN architecture using the "trial and error" procedure by changing the number of layers and hidden neurons, or exploring new optimization control algorithms to achieve the proposed control objectives.

4.2.2. NN Table Models

Neural network tables can take various forms depending on the specific application and the type of data being processed. The most usual NN tables built to analyze NN performance using evaluation metrics include the confusion matrix, the weight matrix, activation tables, and loss tables. In the following, we examine these particular tables.

1. **Confusion Matrix**: This type of table is used to evaluate the performance of a classification model. It shows the numbers of true positives, true negatives, false positives, and false negatives. A generic example is presented as follows:

Actual/Predicted	Positive	Negative
Actual	60	20
Predicted	10	25

2. **Weight Matrix**: In each NN, the weights are critical parameters that transform input data within the network. A weight matrix represents the weights between layers of neurons, for which a simple matrix might look like this:

Neuron k	Neuron k + 1	Neuron k + 2
0.2	0.3	−0.1
−0.3	0.5	0.2

3. **Activation Table**: This type of table illustrates the activation values of the neurons in a NN for a given input, as shown below:

Input k	Input k + 1	Neuron k activation	Neuron k + 1 activation
0.4	0.2	0.5	0.5
0.3	0.7	0.2	0.8

4. **Loss Table**: This table tracks the loss values during NN training, a valuable information for understanding how well the model is learning over time; a figurative example is presented below:

Epoch	Training Loss	Validation Loss
1	0.32	0.35
2	0.15	0.2
3	0.25	0.25

These tables are essential for analyzing and understanding the performance and behavior of neural networks in various applications. Appendix B.2 details two DLNN tables, namely the Tables A1 and A2 with parameter simulation and monitoring training for the two proposed RL DLNN control algorithms developed in Section 4.3. Since both RL DLNN algorithms solve a tracking reference control regression problem, the most suitable evaluation metrics are RMSE and MSE [17]. The strong dependence of reward function of RMSE and MSE gives us the possibility of using an evaluation metric based on reward function, since the problem of minimization of RMSE or MSE to the smallest possible value closest to zero becomes a maximization problem of reward function, for which the most appropriate metrics are the cumulative reward value (return value) or average reward value. The evaluation metric used for DLNN performance in both NN tables is the average reward value for each training episode, since this value appears automatically in the monitoring of the RL agents' training progress, as shown in Figure 9k,l and Tables A1 and A2 from Appendix B.2. The RL agents are trained using the adopted RL DLNNs control algorithms whose reward functions are generated from MPC and the step response specifications of the MIMO Centrifugal Chiller System discussed below in Section 4.3.

4.3. Reinforcement Learning Deep Learning Neural Networks Closed-Loop Control Strategies Applied to a Centrifugal Chiller System

In this section, we develop two advanced intelligent reinforcement learning deep learning neural network (RLDLNN) control algorithms to control the evaporator temperature and the level of refrigerant in the condenser. The first algorithm generates a reward function for an RL Agent based on model predictive control (MPC) specifications and the second algorithm generates two reward functions for two RL agents (controllers) based on the step response block specifications.

4.3.1. Generate Reward Function from the Cost and Constraint Specifications Defined in an MPC Object

This intelligent control strategy and the training phase results were implemented in MATLAB Simulink in Figure 9.

Its compact and detailed architecture is presented in Figure 9a–d and is of practical interest. It automatically generates the reward function from the cost and constraint

specifications defined in a Simulink model predictive controller (MPC) object, as shown in Figure 9e. It is connected in a closed loop with the MIMO centrifugal cooling plant model given by Equations (6) and (7) in Section 2.2 to control the chilled water temperature Tchw-sp in the evaporator subsystem and the level of liquid refrigerant in the second condenser subsystem, depicted in the state-space representation shown in Figure 9g. Implementing this control strategy follows the same procedure that is well described in the updated MathWorks documentation for MIMO Centrifugal Chiller System (CCS) model [18,19]. The MPC object appearing at the top of Figure 9b,c with its parameters block depicted in Figure 9e was created based on the CCS MIMO model, using an inline MATLAB code procedure. For the CCS MIMO plant model under investigation, the sampling time value is set to Ts = 0.1 s, the prediction horizon was assumed to be p = 10 steps, and the control horizon m = 2 steps.

The following specifications of the MPC object are helpful for RL DLNN MPC strategy design:

- Standard linear bounds for output variables (OVs) and manipulated variables (MVs):

$y_{min} = [0\ 0]$, $y_{max} = [10\ 52]$, $mv_{min} = 0$, $mv_{max} = 1.1$, $mv_{ratemin} = -1000$, and $mv_{ratemax} = \text{Inf}$

- Scale factors as specified for OVs and MVs:

$$S_y = [10\ 52],\ S_{mv} = [1.1\ 1]$$

- Standard cost weights:

$$Q_y = [0.1\ 0.1],\ Q_{mv} = [0\ 0],\ \text{and}\ Q_{mvrate} = [0.1\ 0.1]$$

The reward function generated based on the MPC object specifications was the starting point for the reward design. The reward function is used to train the RL Agent, as shown in Figure 9f. The implementer can modify the reward function with different penalty function options and adjust the weights:

The reward function requires the following two components to be computed:

1. The cost component, calculated according to the following equations:

$$dy = (refy(:) - y(:))./{S_y}^T \tag{29}$$

$$dmv = (refmv(:) - mv(:))./{S_{mv}}^T \tag{30}$$

$$dmvrate = (mv(:) - lastmv(:))./{S_{mv}}^T \tag{31}$$

$$J_y = dy^T diag(Q_y.\hat{}2)dy \tag{32}$$

$$J_{mv} = dmv^T diag(Q_{mv}.\hat{}2)dmv \tag{33}$$

$$J_{mvrate} = dmvrate^T diag(Q_{mvrate}.\hat{}2)dmvrate \tag{34}$$

$$Cost = J_y + J_{mv} + J_{mvrate} \tag{35}$$

2. The penalty component for violation of linear bound constraints, with the following components:
   - Penalty function weight (specify nonnegative):

$$W_y = [1\ 1],\ W_{mv} = [10\ 10],\ W_{mvrate} = [10\ 10] \tag{36}$$

     - Choose the **step** or **quadratic** penalty method to calculate the **exteriorPenalty**;

- Set the Pmv value to 0 if the RL Agent action specification has appropriate "*LowerLimit*" and "*UpperLimit*" values.

- Penalty functions:

$$P_y = W_y\left(exteriorPenalty\left(y, y_{min}, y_{max}, '\,\textbf{\textit{step}}'\right)\right) \tag{37}$$

$$P_{mv} = W_{mv}\left(exteriorPenalty\left(m_v, m_{vmin}, m_{vmax}, '\,\textbf{\textit{step}}'\right)\right) \tag{38}$$

$$P_{mvrate} = W_{mvrate}\left(exteriorPenalty\left(m_v - lastmv, m_{vratemin}, m_{vratemax}, '\,\textbf{\textit{step}}'\right)\right) \tag{39}$$

and finally:

$$Penalty = P_y + P_{mv} + P_{mvrate} \tag{40}$$

To calculate the reward value as a result of the MATLAB function block illustrated in Figure 9h, the following relationship between all three components—reward, cost, and penalty—is used:

$$reward = -(Cost + Penalty) \tag{41}$$

For the actual control application under research, the specifications of the RL DLNN environment were the following:

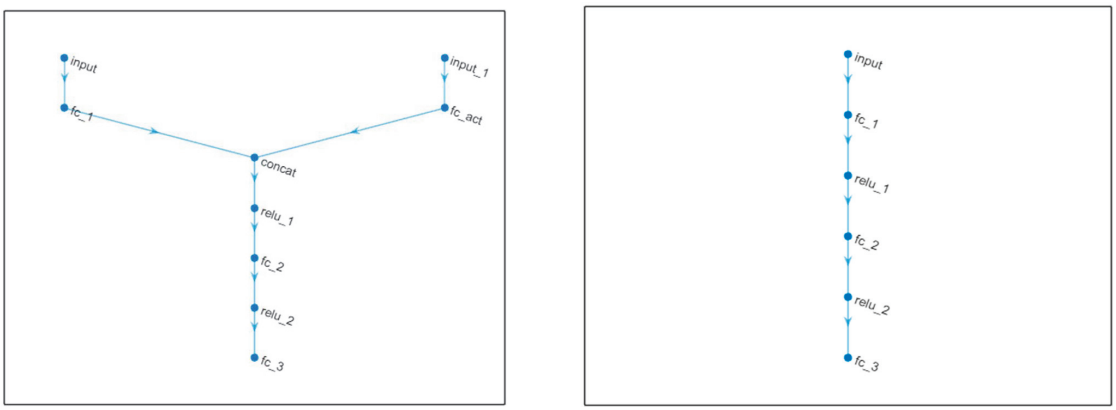
- The observations reference signals (Y_{1sp} and Y_{2sp}), output variables (T_{chw} and *Level*), and their integrals, were as shown in Figure 9j;
- The T_{chw} and *Level* signals were normalized by multiplying with the gain [1/10 1/52];
- The action of U_{com} and u_{EXV} was limited to between [0 1.1] for U_{com} and [0 1] for u_{EXV};
- The sample time and total simulation time were T_s = 0.1 s. In order to capture the full evolution of the dynamics for both plant outputs, the simulation time was set to Tsim = 60 s.

The MATLAB Simulink simulations result is presented in Figure 9.

The block parameters and the block diagram for simulation of reinforcement learning of the RL Agent using a Simulink model as a training and simulation environment are depicted in Figure 9f. The same block also generates the RL Agent greedy policy $\pi_0(.)$, as shown in Figure 9i. Figure 9h shows a screen capture of the MATLAB reward function, and Figure 9j presents a detail of the observation block that appears in Figure 9b,c.

The RL agent chosen in the case study was a twin-delayed deep deterministic policy gradient (TD3) agent [20,21] that used two parametrized Q-value function approximators to estimate the value (that is, the expected cumulative long-term reward) of the policy, as shown in Figure 9f,i. To model the parametrized Q-value function, a neural network that was used had two inputs (the observation and action) and one output corresponding to the value of the policy $\pi_0(\,)$ when taking a given action from the state corresponding to a given observation, as shown in Figure 9i. Before training the RL Agent, each network path of the DLNN was defined as an array of layer objects, and names assigned to the input and output layers of each path in order to connect the paths. Thus, a layer graph object was created and layers aded to generate the criticNet, as is shown in Figure 10a. The critic function objects were created using a dedicated MATLAB *rlQValueFunction* command. To make sure the critics had different initial weights, each network was explicitly initialized before being used create critic 1 and critic 2 [20,21]. The TD3 agent learns a parametrized deterministic policy over continuous action spaces. The policy is learned by a continuous deterministic actor that takes the current observation as input and returns as output an action that is a deterministic function of the observation. The neural network that is used to model the parametrized policy within the actor has one input layer (which receives the content of the environment observation channel) and one output layer (which returns the action to

the environment action channel) [20,21]. An actor network named actorNet is created; its layer graph is shown in Figure 10b. The deterministic actor function is generated for the purposes of modeling the policy of the RL Agent. A set of agent options is specified to train the agent from an experience buffer of maximum capacity 1×10^6 by randomly selecting mini-batches of size 256. It has been reported that a discount factor of 0.995 favors long-term rewards [20,21]. The optimizer options are specified for the actor and critic functions: a learning rate of 1×10^{-3} and a gradient threshold of 1, as set out in [20,21]. During training, the agent explores the action space using a Gaussian noise model of action. The standard deviation and decay rate of the noise are set by using an ExplorationModel property, as is shown in [20,21]. The RL Agent was trained using *train* function, as depicted in Figure 9k; after 13 epochs, it is produced the best tracking performance for chilled water temperature inside the evaporator and for liquid refrigerant level within the condenser. Figure 9l also illustrates the results of the training process after 262 epochs, when the tracking performance of the liquid refrigerant level in condenser was at its best, while that for the chilled water temperature inside the evaporator was slightly attenuated compared with the first result obtained, as shown in Figure 9k. A snapshot of the main steps of the MPC RL DLNN Algorithm A1 is presented in Appendix B.1.



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

Figure 10. Layer graphs for (**a**) criticNet; (**b**) actorNet.

The layer graphs of criticNet and actorNet are displayed in Figures 10a,b, respectively.

4.3.2. Reinforcement Learning Deep Learning Neural Network Control Strategy—Generate Reward Function from a Step Response Specifications of MIMO Centrifugal Chiller Simplified Model in State-Space Representation

Both The digital PID and RL DLNN closed-loop control strategies are depicted in Figure 11a in a compact Simulink diagram architecture. A description in state space of MIMO CCS model is provided in Figure 11b. Figure 11c shows the block parameters of RL Agent1, and Figure 11d details the observations block as a component of the same compact Simulink diagram. The compact Simulink diagram also shows how to automatically generate two reward functions—rewardfunctionVfb1 and rewardfunctionVfb2, shown in Figure 11e,g from the performance requirements defined in the Simulink Design Optimization model verification blocks provided in MATLAB R2024a Simulink Toolbox for evaporator temperature control (chosen as the first agent, RL Agent1) and condenser liquid refrigerant level control (the second agent, RL Agent2), respectively. Also in the compact Simulink diagram appear two Boolean logic termination blocks, which stop the training of the RL Agent1 and RL Agent2 when the evaporator temperature and the condenser level reach good accuracy performance, as shown in Figure 11f,h. The rewardfunctionVfb1 and rewardfunctionVfb2 are used to train both reinforcement learning RL Agent1 and RL

Agent2, following the stated steps and the MATLAB Reinforcement Learning subroutine procedure similar to those developed in [20,21] applied for MIMO plants. The training of both agents, RL Agent1 and RL Agent2, is performed similarly to the MPC RL DLNN procedure described above in Section 4.3.1. Compared with MPC RL DLNN, both RL agents are decentralized with two distinct paths. The untrained simulation results for evaporator temperature control and condenser liquid refrigerant level control are presented in Figure 11i,j, respectively. Figure 11k,l presents the training simulation results for the evaporator temperature and liquid refrigerant level inside of the condenser after 200/200 epochs. As shown in Figure 11m, the training process of both RL Agent1 and RL Agent2 finishes after the agents reach the criteria to stop training set out in the training options MATLAB subroutine presented in the print screen snapshot of Algorithm A2 from Appendix B.1. The penalty function weight is specified nonnegative and set to the value 2. To compute the penalty for violation of linear bound constraints, the same MATLAB functions were used as described in previous section for MPC strategy. The user can specify the penalty method as step or quadratic; the quadratic is typically preferred.

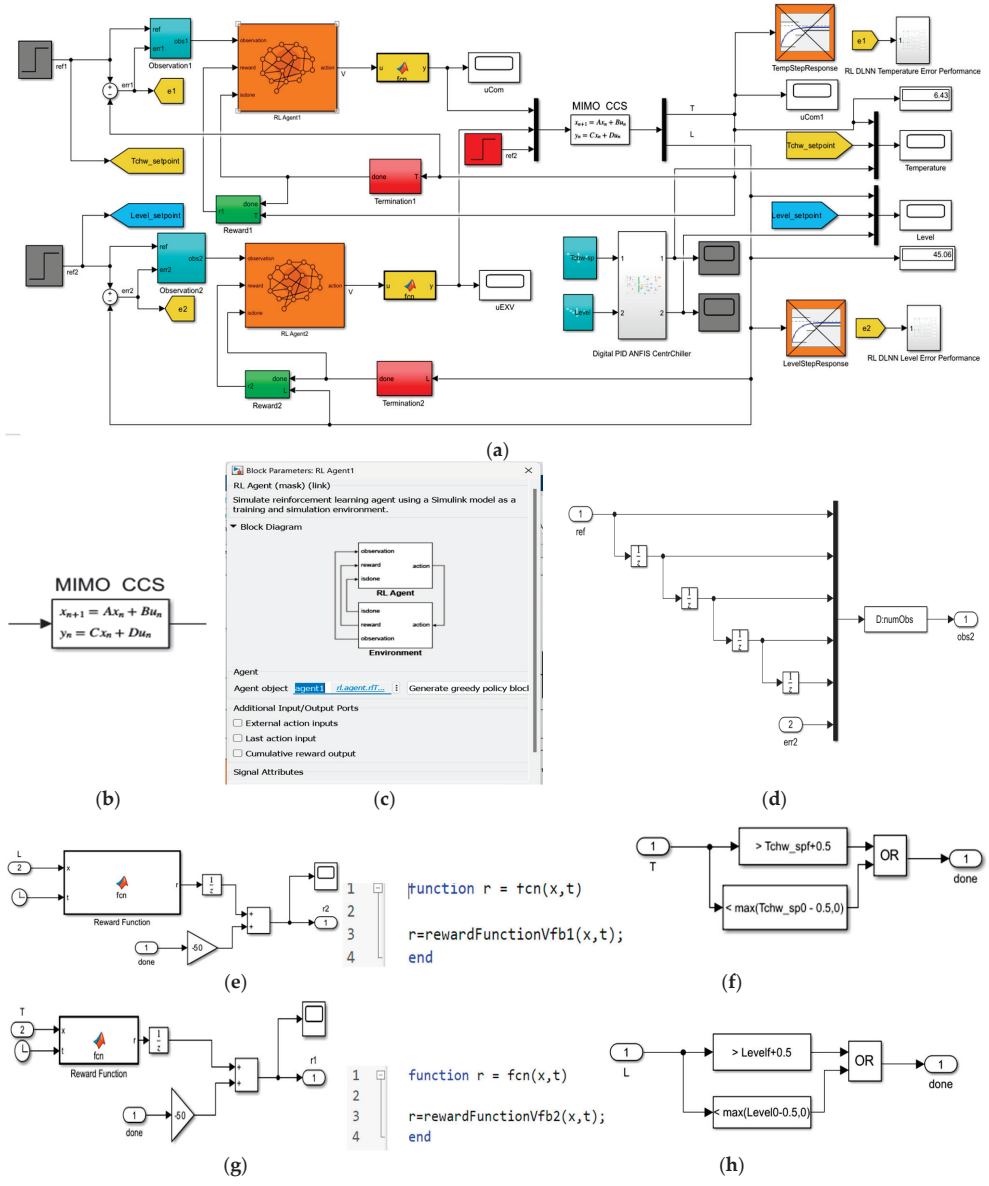
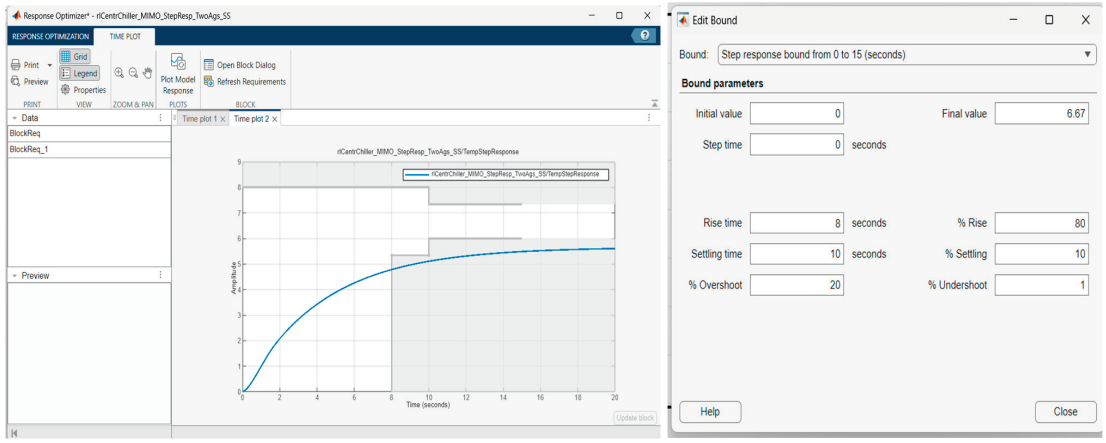
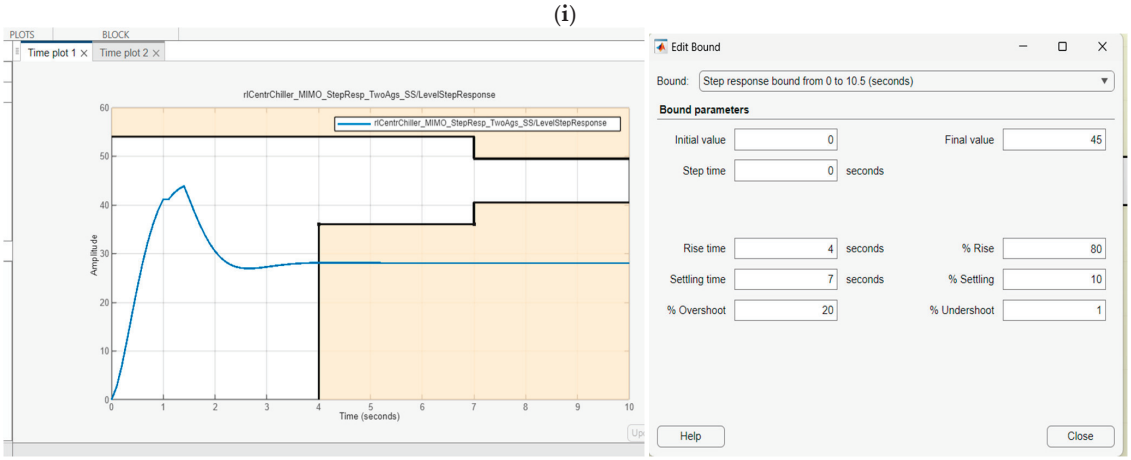


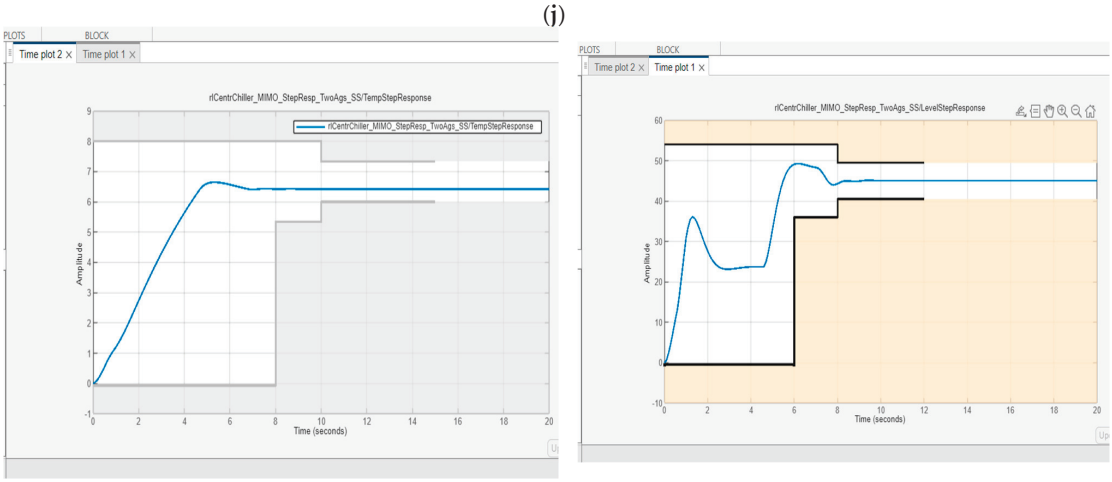
Figure 11. *Cont.*



(**i**)



(**j**)



(**k**) (**l**)

Figure 11. *Cont.*

(**m**)

Figure 11. Reinforcement learning deep learning neural network Simulink diagram: (**a**) overall Simulink diagram of RL DLNN control architecture and MIMO CCS; (**b**) MIMO CCS state-space model; (**c**) RL Agent1 block; (**d**) observation block; (**e**) Reward Function Vfb1 block; (**f**) termination block for first RL Agent1; (**g**) Reward Function Vfb2 block; (**h**) termination block for first RL Agent2; (**i**) chilled water temperature within evaporator and step response block specifications-untrained RL1 Agent; (**j**) liquid refrigerant level inside condenser and step response block specifications-untrained RL Agent2; (**k**) chilled water temperature within evaporator-trained RL1 Agent after 33/200 epochs; (**l**) liquid refrigerant level inside condenser-trained RL2 Agent after 33/200 epochs; (**m**) training process ofRL1 and RL2 Agents.

The corresponding MATLAB code lines inside the MATLAB blocks built based on the step response specifications for each reward function are presented in the same Algorithm A2 from Appendix B.1. The MATLAB Simulink simulations results are shown in Section 6.2, below.

5. Traditional and Advanced Intelligent Closed-Loop Control Strategies—MATLAB Simulink Simulation Results

This section presents the MATLAB Simulink simulation results of traditional closed-loop control strategies. Section 5.1.1 discusses DTI control of the MIMO CCS, and in Section 5.1.2 is depicted the simulations result for a PID control of MIMO CCS extended nonlinear model with 39 states, inputs subjected to constraints, and under measured temperature disturbance. In Section 5.1.3 are shown the simulation results of an improved version of PID control, more precisely, a digital PID control of a MIMO CCS ANFIS model. Model predictive control of the MIMO CCS model represented in state space with four states is discussed in Section 5.1.4. Also, the reinforcement learning MPC deep learning neural network (RL DLNN) control MIMO CCS model is compared versus MPC control in Section 5.2.1, and the RL DLNN control of MIMO CCS model in state-space representation

versus the improved digital PID control of MIMO CCS ANFIS model is compared in Section 5.2.2.

5.1. Traditional Closed-Loop Control Strategies

5.1.1. DTI Closed-Loop Control

The simulation results are presented in Figure A2 in Appendix A. Figure A2a shows the Simulink diagram of the DTI controller. Figure A2b depicts the DTI control of chilled water temperature control within the evaporator subsystem, and Figure A2c the liquid refrigerant level control inside the condenser subsystem. Figure A2d presents the compressor and expansion valve opening actuator control efforts.

5.1.2. PID Closed-Loop Control—Centrifugal Chiller Extended Model (39 States)

The Simulink simulation result is depicted in Figure 12. Figure 12a presents the PID MIMO Centrifugal Chiller closed-loop temperature control inside the evaporator subsystem; in Figure 12b, the results for PID control of the liquid refrigerant level in the condenser subsystem are revealed. Figure 12c depicts the compressor actuator control effort and Figure 12d discloses the expansion valve opening actuator control effort.

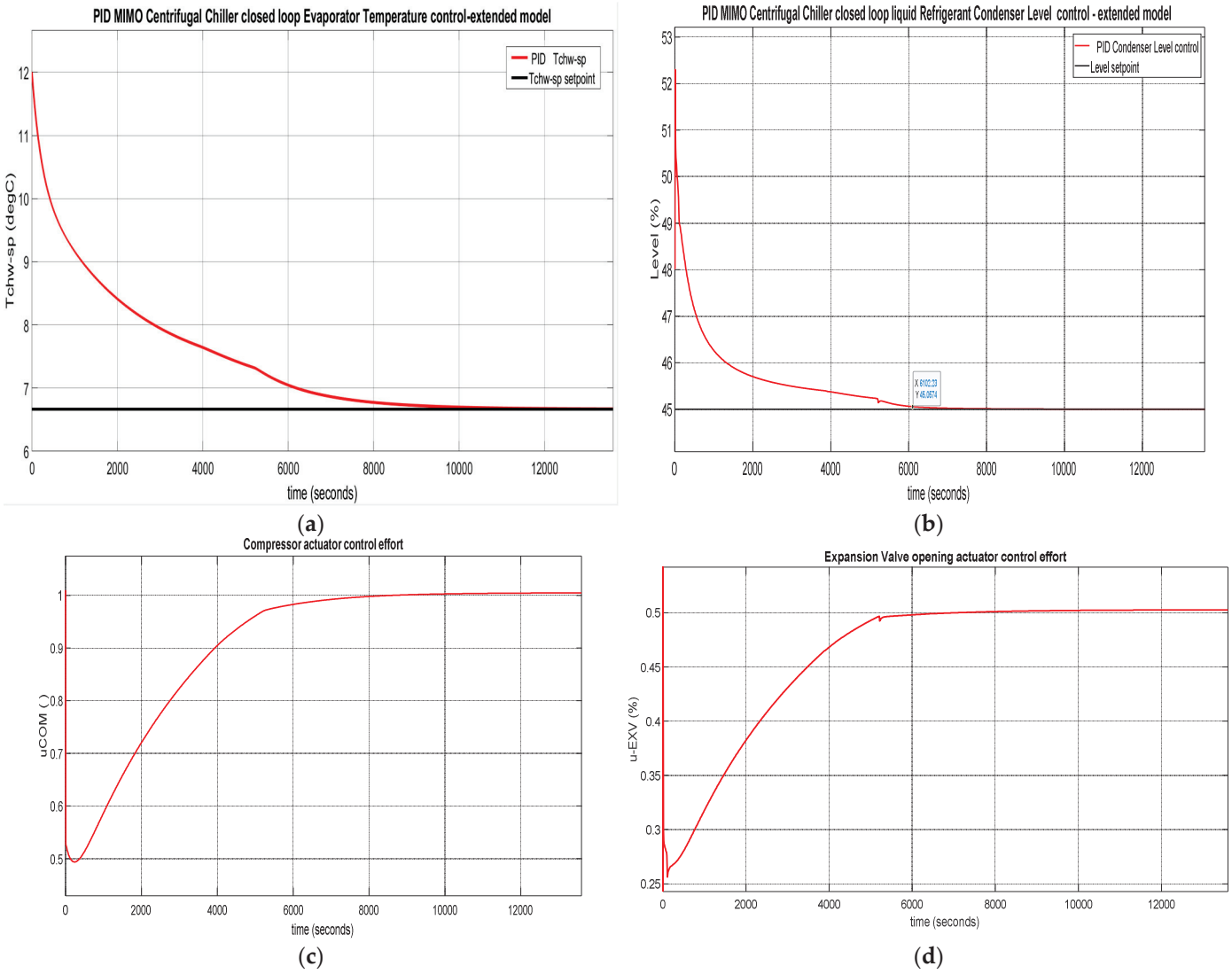


Figure 12. MATLAB Simulink simulation results: (**a**) PID evaporator subsystem temperature control; (**b**) PID liquid refrigerant level control in condenser subsystem; (**c**) compressor relative speed actuator control effort; (**d**) expansion valve opening actuator control effort.

5.1.3. Digital PID Control of MIMO Centrifugal Chiller ANFIS Model

The MATLAB Simulink simulations results are revealed in Figures 13 and 14, and also in Figures A4–A7 in Appendix A. The MATLAB Simulink simulation results for the digital PID control CCS ANFIS model without changes in temperature and level setpoints are depicted in Figure 13a,b, respectively. The actuators' control efforts for the compressor and expansion valve opening are shown in Figure 14a,b. On simple visual inspection of both figures, it seems that the fast digital PID control had a fast step response, reaching zero steady-state for a settling time of 50 s with a 9% overshoot for chilled water inside the evaporator and 2% for liquid refrigerant level within the condenser.

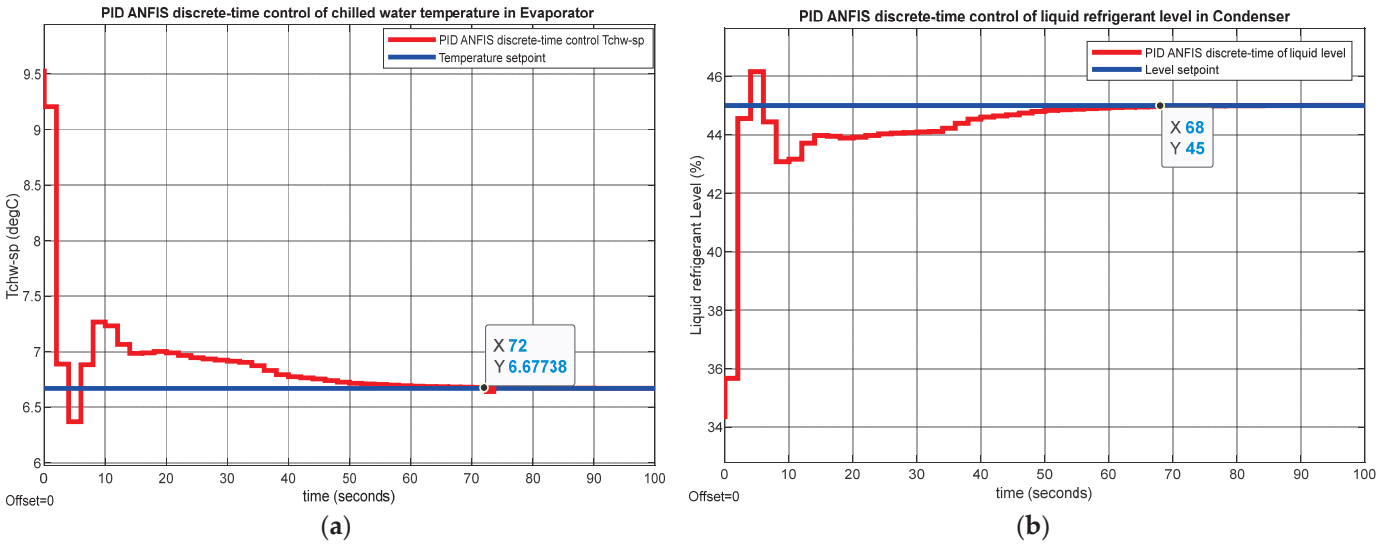


Figure 13. MATLAB Simulink simulation results without changes in temperature and level setpoints: (**a**) PID ANFIS discrete-time control of Tchw-sp in evaporator; (**b**) PID ANFIS discrete-time control of liquid refrigerant level in condenser.

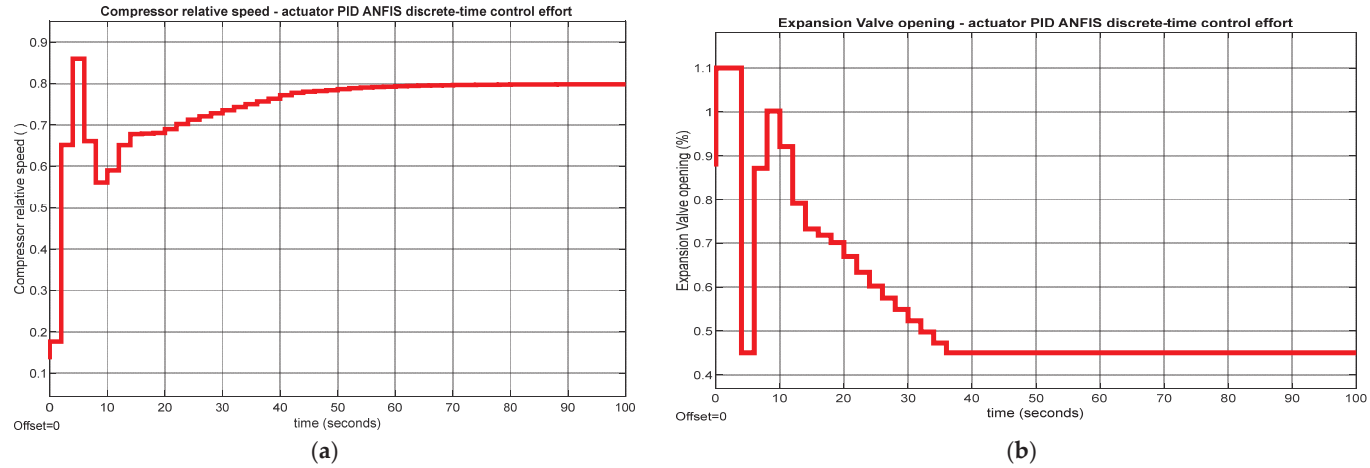


Figure 14. PID ANFIS discrete-time control—MATLAB Simulink simulation results of actuator control efforts: (**a**) compressor relative speed; (**b**) expansion valve opening.

The ability of the digital PID controller to overcome the effects of temperature disturbance is revealed in Figure A4a for temperature and in Figure A4b for liquid refrigerant level, respectively. Figure A5a shows the impact of temperature disturbance on the compressor actuator control effort, and Figure A5b on expansion valve opening actuator effort, respectively. Figure A6a shows the evolution of the chilled water temperature inside the evaporator subsystem, and Figure A6b shows the evolution of the liquid refrigerant level

in the condenser subsystem with changes in setpoint values for temperature and level, respectively. The impact of these setpoint changes on the compressor and expansion valve opening actuator control efforts are shown in Figure A7a,b, respectively.

5.1.4. Model Predictive Control of MIMO Centrifugal Chiller Nonlinear Extended Model in State-Space Representation (39 States) with Input Constraints

The MPC simulation results are described in Figure 15a–d. The control effects of both actuators, compressor relative speed and expansion valve opening, are represented in Figure 15a. Figure 15b reveals the chilled water temperature disturbance, Trr. Figure 15c shows the MPC of chilled water temperature (OV1) inside the evaporator in degrees Celsius, and Figure 15d displays the liquid refrigerant level (OV2) within the condenser with a change in temperature disturbance Trr from 48 [degF] to 54 [degF] applied at time instant t = 40 [s].

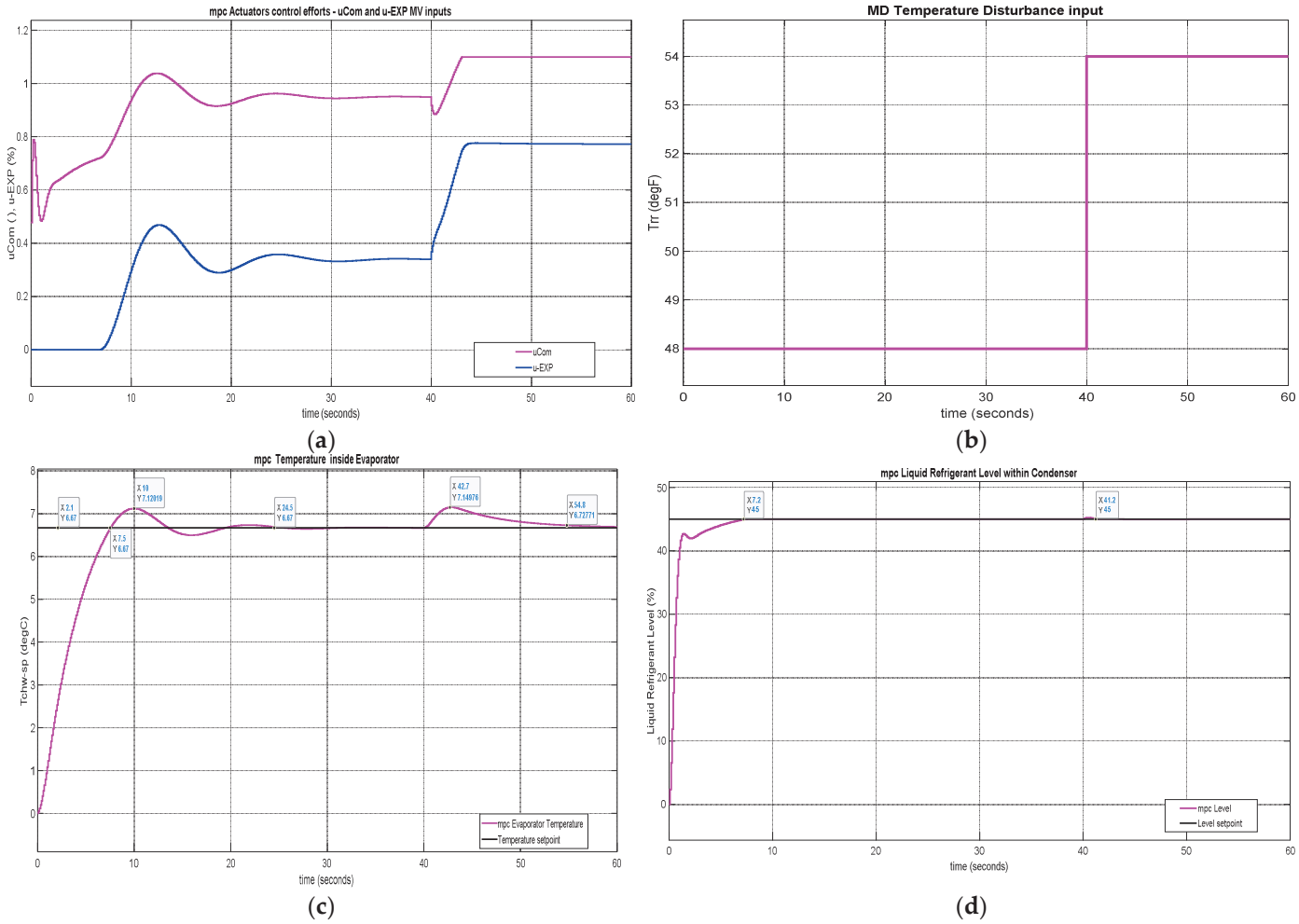


Figure 15. MPC MATLAB Simulink simulations resulted in a closed loop for input (u1 = Ucom, u2 = uEXV, u3 = Trr) and output (y1 = Tch-sp, y2 = Level) responses against internal plant using mpcDesigner application based on the input–output measurement dataset for the MIMO CCS extended state model: (**a**) actuator control effort; (**b**) temperature disturbance; (**c**) chilled water temperature in evaporator; (**d**) liquid refrigerant level in condenser.

5.2. Advanced Reinforcement Learning Using Deep Learning Neural Network Control Strategies

This subsection presents the simulation results of the advanced reinforcement learning deep learning neural networks (RL DLNNs) developed in Section 4, conducted on the MATLAB Simulink R2024 software programming platform.

The Section 5.2.1 refers to the Simulink simulation results for the RL DLNN that generated the reward function from an MPC dataset extracted from the MIMO CCS simplified model represented in state-space, given in Equation (7). Section 5.2.2 presents the simulations result for the RL DLNN that generated the reward function from two step responses' block specifications for the same MIMO CCS simplified model in state-space representation.

5.2.1. Reinforcement Learning Deep Learning Neural Network Control Strategies–Generate Reward Function from MPC of MIMO Centrifugal Chiller Simplified Model in State-Space Representation

The Simulink simulations results are shown in Figure 16a–c. Results for the RL DLNN control of evaporator chilled water temperature are displayed in Figure 16a, and Figure 16b represents the liquid refrigerant level in the condenser, both versus their corresponding MPC step responses on the same graphs. The Figure 16c illustrates the separate RL DLL MPC control efforts.

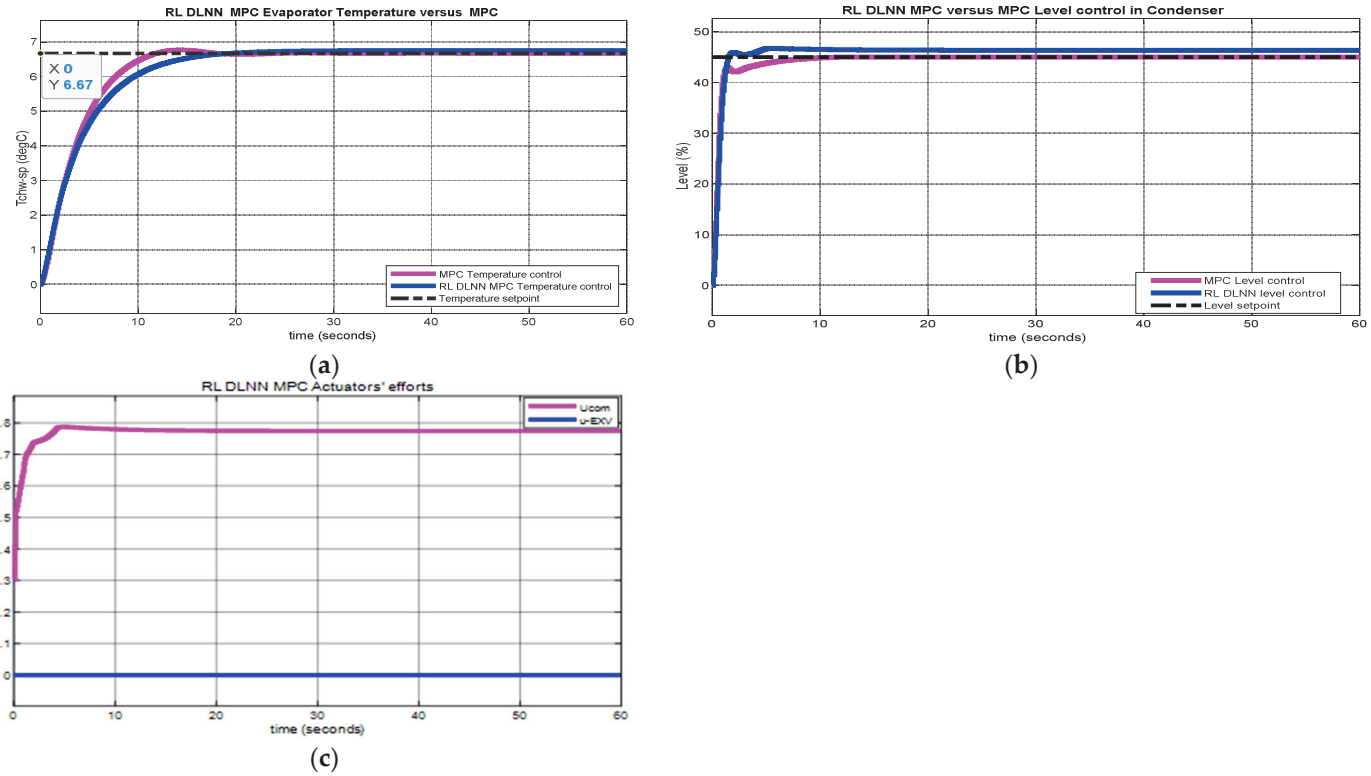


Figure 16. RL DLNN—Generate reward function based on MPC specifications: (**a**) RL DLNN MPC evaporator temperature control versus MPC; (**b**) RL DLNN MPC condenser level control versus MPC; (**c**) RL DLNN MPC actuator efforts.

5.2.2. Reinforcement Learning Deep Learning Neural Network Control Strategies—Generate Reward Function from Step Response Specifications of a MIMO Centrifugal Chiller Simplified Model in State-Space Representation

The simulation results for RL DLNN control of evaporator chilled water temperature are depicted in Figure 17a, and Figure 17b illustrated the liquid refrigerant levels versus digital PID control of MIMO ANFIS model responses.

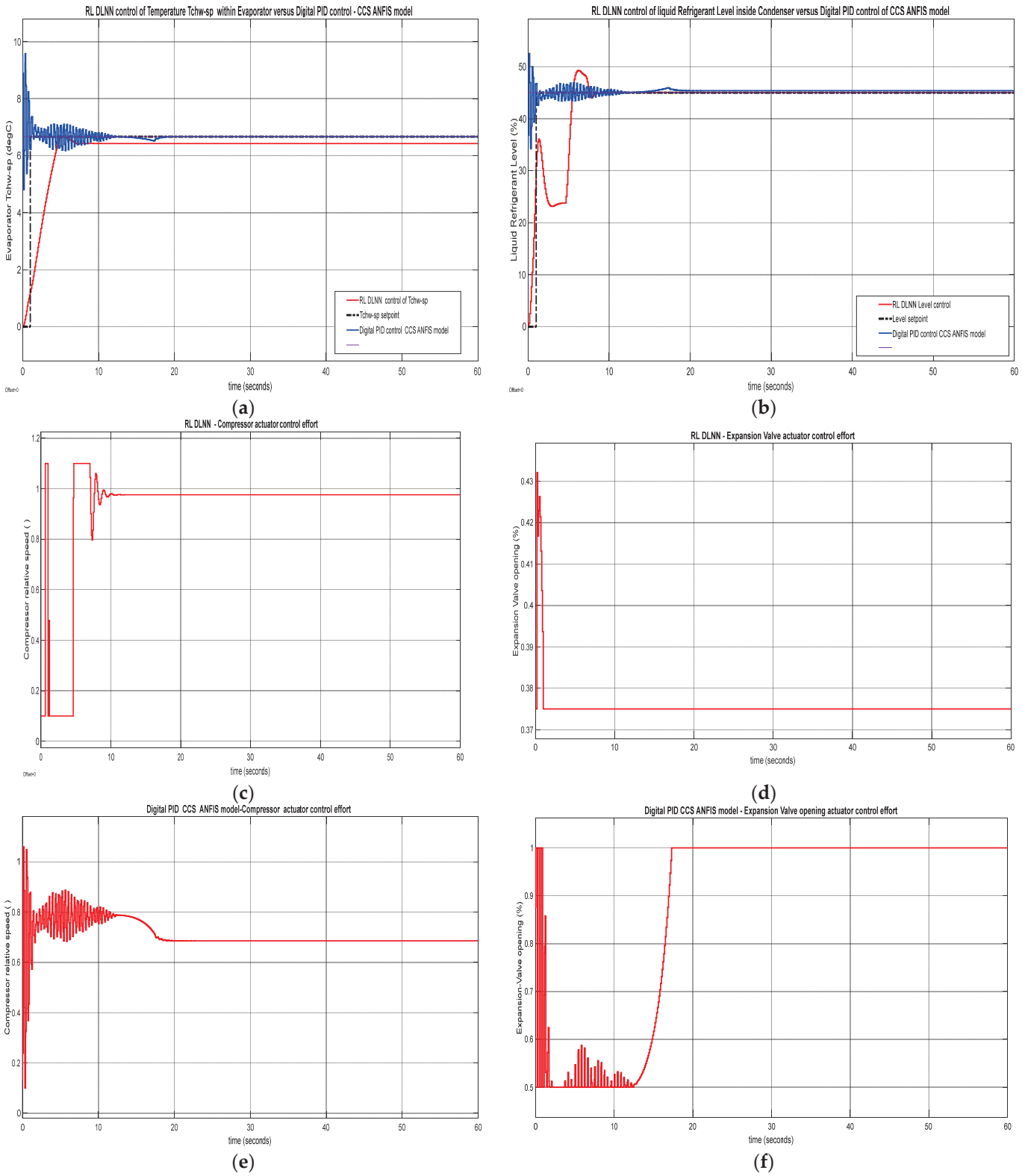


Figure 17. MATLAB Simulink simulation results of RL DLNN versus digital PID MIMO ANFIS model generate reward function from the step response specifications of the MIMO Centrifugal Chiller simplified model in state-space representation. (**a**) RL DLNN of chilled water temperature inside the evaporator versus digital PID controller using ANFIS CCS models; (**b**) RL DLNN control of liquid refrigerant level within the condenser versus digital PID controller using ANFIS CCS models; (**c**) compressor actuator control effort, RL DLNN; (**d**) expansion valve opening actuator control effort, RL DLNN; (**e**) compressor actuator control effort, digital PID CCS ANFIS model; (**f**) expansion valve opening actuator control effort, digital PID CCS ANFIS model.

The RL DLNN actuators control efforts for the compressor and for the expansion valve opening is illustrated in Figure 17c,d, respectively, while the control efforts of digital PID control CCS ANFIS models are shown in Figure 17e for the compressor actuator and Figure 17f for expansion valve opening actuator.

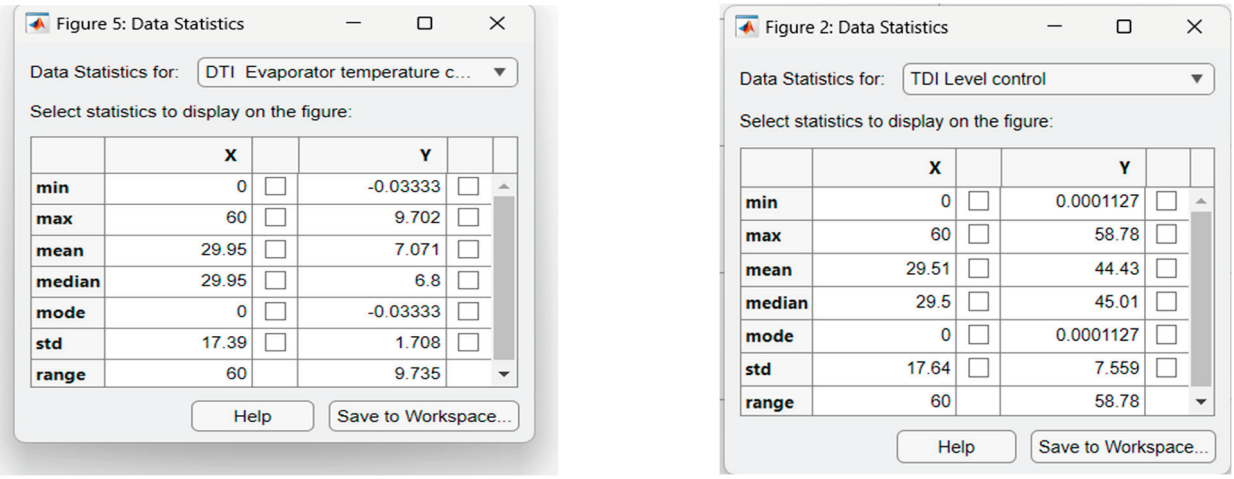
6. Discussion

6.1. Conventional Control Strategies

In this section, a rigorous performance analysis is presented based on the statistical data and stability performance error indicators extracted from step responses and the standard structures used to calculate these indicators, such as ISE, ITSE, IE, IAE, and ITAE.

6.1.1. DTI Controllers

For the discrete-time integrator (DTI) controller developed in Section 3.1, through a simple inspection of both the step responses represented in Figure A2a,b and the statistics extracted for the same step responses for evaporator temperature, depicted in Figure 18a, and liquid refrigerant level inside the condenser, reported in Figure 18b, the following performances during transient and steady state regimes were calculated: acceptable time responses (settling time, Ts) and rise time (Tr) for temperature Ts = 40, Tr = 5.5 s and for liquid refrigerant level Ts = 20, Tr = 1.7 s, an overshoot of σ_max = 45.27% for temperature, and a larger number of maximum amplitude oscillations, of about 10%, both reaching zero steady-state errors, therefore showing high tracking accuracy.



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

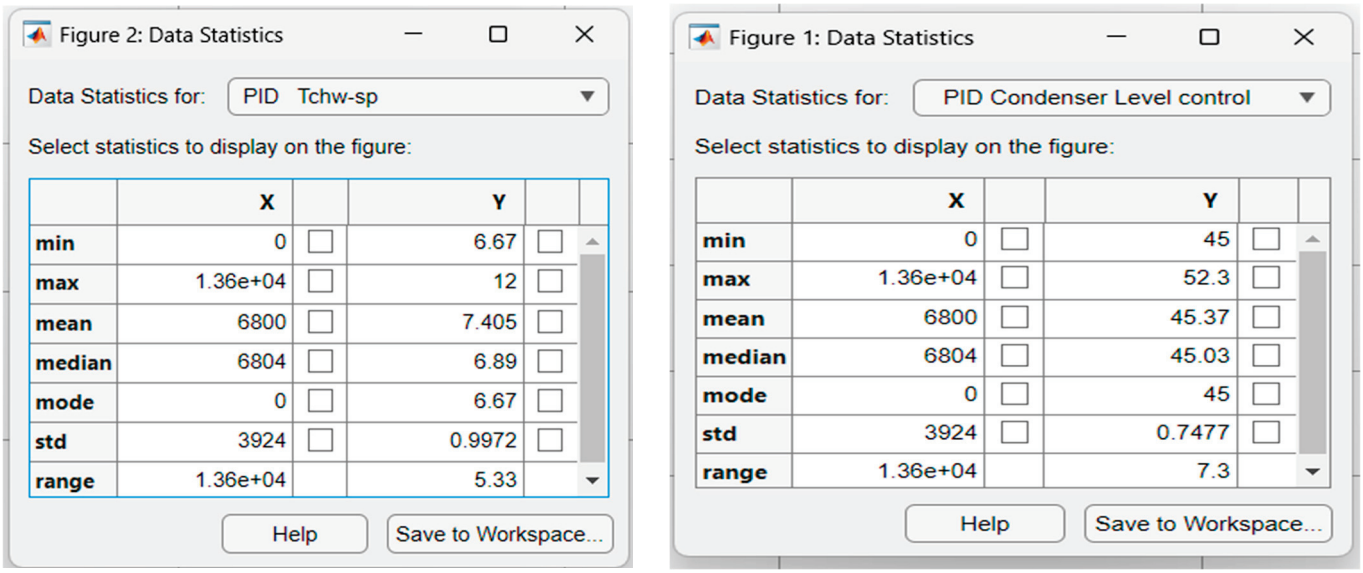
Figure 18. Statistical data extracted from the time responses: (**a**) TDI evaporator temperature control; (**b**) TDI condenser liquid refrigerant level control.

Mean = 7.071 and standard deviation (std) = 1.708 for DTI temperature control can be extracted from Figure 18a, while mean = 44.43 and std = 7.759 for DTI liquid refrigerant level control is given in Figure 18b. It is also worth noting the strong control effort of the oscillating compressor and the almost smooth opening of the expansion valve.

6.1.2. PID Control MIMO Centrifugal Chiller System Extended Model with 39 States

The step responses shown in Figure 12a,b reveal a very long Ts = 10,000 s, Tr = 10,000 s, no overshoot, zero steady-state error, smooth compressor control effort for PID temperature control, and a very long Ts = 6000, Tr = 6000 s, no overshoot, zero steady-state error, and smooth expansion valve opening control effort for PID level control. The main issues encountered for this closed-loop control strategy were a very slow time response and the need for accurate tuning of the parameter values. Also, mean = 7.405 and std = 0.9972 for

PID temperature control are revealed in Figure 19a, and mean = 45.03 and std = 0.7477 for PID level control are shown in Figure 19b.



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

Figure 19. Statistical data extracted from the time responses: (**a**) PID evaporator temperature control; (**b**) PID condenser liquid refrigerant level control.

6.1.3. Model Predictive Control of MIMO Centrifugal Chiller Simplified Model in State-Space Representation

The rigorous performance analysis of step responses depicted in Figure 15c for the evaporator temperature subsystem and in Figure 15d for the liquid refrigerant level in the condenser subsystem revealed the following features:

- settling time Ts = 24.6 s, rising time Tr = 7.6 s, an overshoot of σ_max = 6.75%, zero steady-state error, and an excellent disturbance rejection for evaporator temperature control;
- settling time Ts = 7.2 s, rising time Tr = 7.2 s, no overshoot, a high tracking performance accuracy, and significant disturbance rejection for the liquid refrigerant level inside the condenser subsystem.

These results recommend the MPC closed-loop strategy as among the most suitable conventional control strategies. It performs very well, and neither the compressor relative speed and nor the expansion valve opening input violated the linear bound constraints. The MPC adopted in the first RL DLNN closed-loop control strategy was developed in Section 4.3.1 to generate the reward function from the MPC specifications of the MIMO Centrifugal Chiller System model proposed in the case study.

6.2. Advanced Intelligent Closed-Loop Neural Control Strategies

Digital PID Control MIMO CCS MISO ANFIS Models

The MATLAB Simulink simulation results of the digital PID control CCS ANFIS model without changes in temperature and level setpoints, depicted in Figure 13a,b, respectively, show a fast time response Ts = 50 s from the evaporator temperature control subsystem, with zero steady-state error and a 9% temperature of chilled water overshoot, compared with the liquid refrigerant level control inside the condenser subsystem which had Ts = 64 s, steady-state error, and 2% overshoot of the liquid refrigerant level in the condenser. Moreover, the actuator control efforts for the compressor and expansion valve opening, correspondingly shown in Figure 14a,b, are sharp at the beginning and smooth later. The

Simulink simulation results depicted in Figure A4a,b reveals that the effects of changes in disturbance temperature load Trr from 48 to 54 degrees Fahrenheit applied at time t = 70 s were completely overcome via the chilled temperature control inside the evaporator, as were those in the level of liquid refrigerant within the condenser, almost within 40 s. Also, the figures indicate that the steady state for each controlled output was very close to zero, and the settling times of the step responses were fast enough for both digital PID controllers, which can be interpreted as excellent performance using the improved digital PID control strategy, outperforming the previous control PID control strategy based on the extended model developed in Section 3.2. For this reason, the digital PID with the proposed tuning procedure for the parameters' values was used for performance comparison with the second RL DLNN control structure.

6.3. Advanced Reinforcement Learning Deep Learning Neural Networks Control Strategies

For a rigorous analysis of tracking performance, it is essential to minimize errors in any closed-loop feedback control system. In order to keep track of errors at all time, from zero to infinity, and to minimize them continuously, performance measures were calculated including integral time absolute error (ITAE), integral square error (ISE), integral time square error (ITSE), and integral absolute error (IAE), as previously defined [15] and given by Equations (42)–(46).

An effective tracking performance measure uses data statistics such as min, max, mean, median, mode, standard deviation (std), and range, which are included in Figures A8a–d and A10a–d. As discussed elsewhere [15], the integral time indicators defined by Equations (42)–(45) are so-called fitness functions. Their values are calculated by the standard hardware structures shown in Figures A9a–d and A11a–d, defined as follows:

$$ISE = \int e(t)^2 dt \tag{42}$$

$$ITSE = \int t\left[e(t)^2\right] dt \tag{43}$$

$$IAE = \int |e(t)| dt \tag{44}$$

$$ITAE = \int t|e(t)| dt \tag{45}$$

$$IE = \int e(t) dt \tag{46}$$

Specifically, all of these time integral criteria are generic and comprehensive tools to evaluate the performance of a control system. Hence, a system may have a good rise time but a poor settling time, or vice versa, while other basic criteria for evaluating the step response tracking performance of a closed-loop feedback system include the overshoot and the steady-state error extracted from the statistical data, although all these statistics describe only one characteristic. The time integral criteria are generic and comprehensive; they allow comparison between different controller designs or even differently structured controllers. The fitness functions are not actually limited to the above equations. Engineers can provide custom fitness functions depending on the target design and control system. The overall performance (convergence speed and optimization accuracy) of an interesting evolutionary algorithm previously developed [15] for optimal tuning of the PID parameter values depends on the fitness functions. Note that integral time absolute error (ITAE) is widely used in control processing since it is simple to implement and to define the energy of signals which demonstrate symmetry and differentiability.

6.3.1. Reinforcement Learning Deep Learning Neural Network Control Strategies—Generating a Reward Function from the MPC of MIMO Centrifugal Chiller Simplified Model in State-Space Representation

The statistical data tables for the RL DLNN temperature and level control algorithms versus MPC of evaporator temperature and condenser refrigerant level are shown in the Figure A8a–d. Figure A8a presents the statistical data for the MPC for evaporator temperature control, Figure A8b for the RL DLNN temperature control, Figure A8c visualizes the data for the condenser refrigerant level, and Figure A8d includes the RL DLNN refrigerant level control data. The performance error indicator values extracted from the hardware structures are given in Figure A9a–d; precisely, for MPC evaporator temperature in Figure A9a, RL DLNN MPC evaporator temperature in Figure A9b, MPC condenser refrigerant level in Figure A9c, and finally, RL DLNN MPC condenser refrigerant in Figure A9d.

Attentive performance analysis based on the data statistics for the MPC RL DLNN evaporator temperature control subsystem revealed a slight superiority compared with MPC based on the corresponding data statistics and vice versa for the condenser liquid level control subsystem. Regarding the final fitness function values after 60 iterations, the Simulink simulations results indicated a slight superiority on the part of the MPC RL DLNN evaporator temperature control subsystem compared with the MPC, and vice versa for the condenser liquid level control subsystem. The MPC RL DLNN performed better for temperature control and similarly for liquid refrigerant level control. After many other repetitions of the parameter values tuning procedure, which was a so-called "trial and error" procedure, the performance of MPC RL DLNN simulation results was observed to change significantly, such that it was finally able to outperform the conventional MPC for control of both temperature and liquid refrigerant level. Our investigations will continue to improve the performance io the MPC RL DLNN controller in future work.

6.3.2. Reinforcement Learning Deep Learning Neural Network Control Strategies—Generating Reward Function from a Step Response Specifications of MIMO Centrifugal Chiller Simplified Model in State Space Representation

Similarly, the statistical data tables for the RL DLNN temperature and level control algorithms versus the digital PID control–CCS ANFIS model of evaporator temperature and condenser refrigerant level are shown in Figure A10a–d. The Figure A10a presents the statistical data for RL DLNN evaporator temperature control, Figure A10b for digital PID temperature control CCS ANFIS model, Figure A10c visualizes the statistics for RL DLNN condenser refrigerant level, and Figure A10d illustrates the digital PID refrigerant level control CCS ANFIS model results. The performance indicator values were extracted from the hardware structures illustrated in Figure A11a–d: for RL DLNN evaporator temperature in Figure A11a, the digital PID evaporator temperature control CSS ANFIS model in Figure A11b, RL DLNN condenser refrigerant level in Figure A11c, and finally the digital PID condenser refrigerant level control CCS ANFIS model in Figure A11d. Similar to MPC RL DLNN, rigorous performance analysis conducted for the RL DLNN based on the statistical data and fitness functions' final values after 60 iterations indicated a slight superiority of the improved digital PID controller connected in the same forward-path closed loop architecture with the MISO ANFIS CCS models, compared with the RL DLNN's performance, with both these controllers outperformed the standard PID controller. Furthermore, by increasing the number of "trial-and-error" procedures for tuning parameter values, it seems certain that the performance of RL DLNN results can improve significantly, until it can finally outperform the improved digital PID control structure for control of both temperature and liquid refrigerant level. Our investigations in future work will continue in this direction.

6.4. Considerations Regarding the RL DLNN Control Strategies' Applicability

The proposed RL DLNN control strategies are well suited for handling the complexities of real-world centrifugal chiller systems, due to the abilities of the RL agents to cope with dynamic environments (e.g., varying cooling loads, weather conditions, or equipment performance) and to detect and compensate for system faults or degradation. Moreover, with the rapid advancements in edge computing and hardware acceleration, RL controllers can perform real-time optimization, making them practical for real-world applications. Moreover, RL DLNN controllers can be effectively scaled for centrifugal chiller systems due to a combination of attributes, including their modularity, ease of transfer learning (i.e., pre-trained RL models can be fine-tuned for different chiller systems or operating conditions, reducing the need for extensive retraining), and distributed control capabilities (i.e., RL agents can be deployed in a distributed manner, controlling multiple chillers in a plant or across different locations, and coordinated to achieve global objectives).

From another point of view, the applicability of the proposed RL DLNN control strategies in real-life scenarios can be affected by three important factors: (a) the customized reward function design based on step response specifications may not cover all operational constraints, such as energy efficiency, or the long-term stability of the MIMO Centrifugal Chiller control system, requiring fine-tuning for each specific HVAC application; (b) safety and stability concerns resulting from possibly damaging actions the RL agents explore during training; and, (c) high computational cost, since the RL DLNN models can be computationally expensive to train and deploy, demanding significant hardware resources. These limitations may be mitigated by employing carefully pre-trained RL agents deployed on real-time and low-cost embedded systems.

7. Conclusions

This study explored the development and implementation of two intelligent neural reinforcement learning control algorithms utilizing deep learning neural network frameworks in the specific case of a complex HVAC centrifugal chiller system characterized by high dimensionality and nonlinearity, strict constraints, and significant impact of measured disturbances. For this, two simplified MIMO models of the CCS were generated, and a comprehensive series of simulations were conducted to showcase the efficacy of both RL DLNN control algorithm implementations when compared to two traditional control methods. The two data-driven advanced neural control algorithms this paper proposes have demonstrated their viability and adaptability to various kinds of nonlinearities, singularities, and uncertainties. In future work, we intend to take a further step by implementing our control strategies in real-life scenarios.

Author Contributions: Conceptualization, N.T.; methodology, D.-I.C. and N.T.; software, D.-I.C. and R.-E.T.; validation, M.Z. and N.T.; formal analysis, R.-E.T.; investigation, N.T. and D.-I.C.; resources, M.S.R.; data curation, M.Z.; writing—original draft preparation, R.-E.T.; writing—review and editing, N.T.; visualization, R.-E.T. and S-M Radu; supervision, N.T. and M.Z.; project administration, M.S.R.; funding acquisition, M.S.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: N. Tudoroiu expresses his gratitude to Roberta Silerova, John Abbott College Program Dean Science, Chair Mark Evanchyna and Co-Chair Hana Chammas of the Intech Department, as well as his departmental colleagues Demartonne Ramos França, Shiwei Huang, and Evgeni Kiriy for their support, guidance, and valuable advice during the course of this scientific research project.

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

Abbreviations

The following abbreviations are used in this manuscript:

PID	Proportional integral derivative
DTI	Discrete time integrator
HVAC	Heating Ventilation Air Conditioning
CCS	Centrifugal Chiller System
MIMO	Multi-input multi-output
SISO	Single-input single-output
MISO	Multi-input single-output
ARMAX	Autoregressive moving average with exogenous input
ANFIS	Adaptive Neural Fuzzy Inference System
IAE	Integral absolute error
ITAE	Integral time absolute error
ISE	Integral square error
ITSE	Integral time-weighted square error

Appendix A

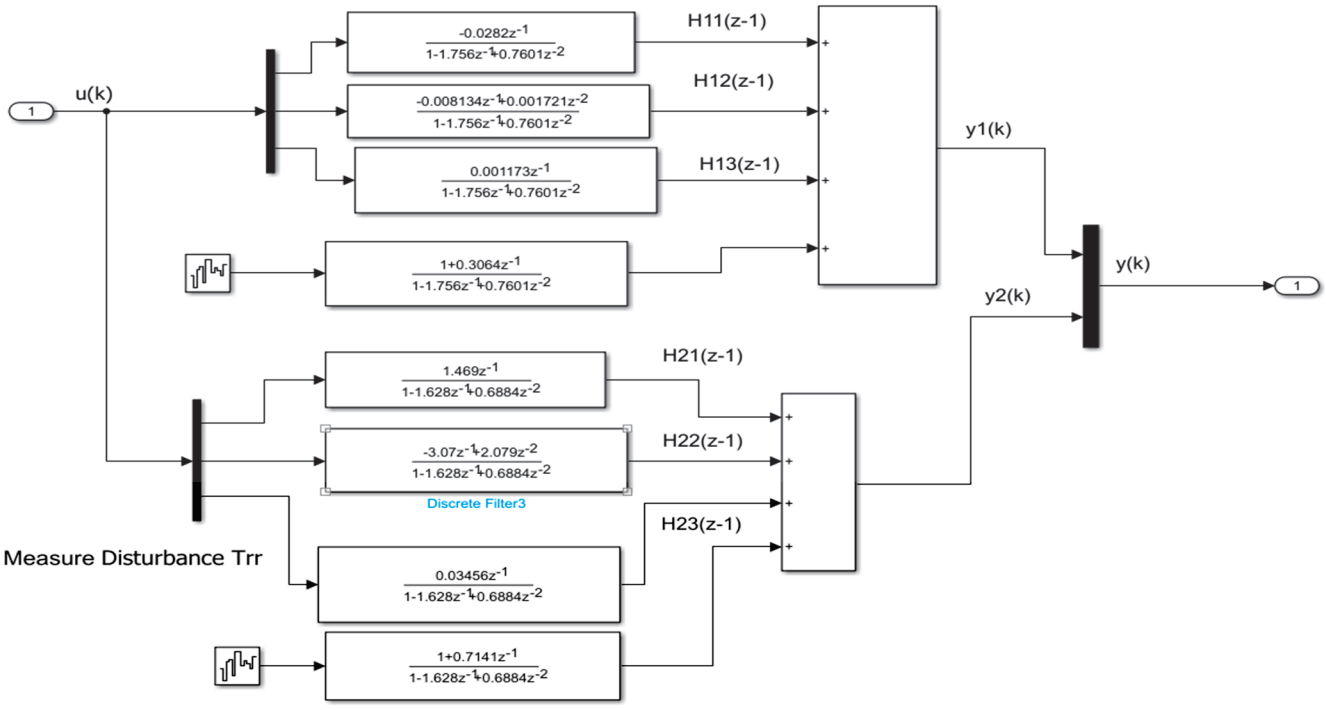


Figure A1. Simulink diagram of discrete-time transfer function of the MIMO Centrifugal Chiller plant.

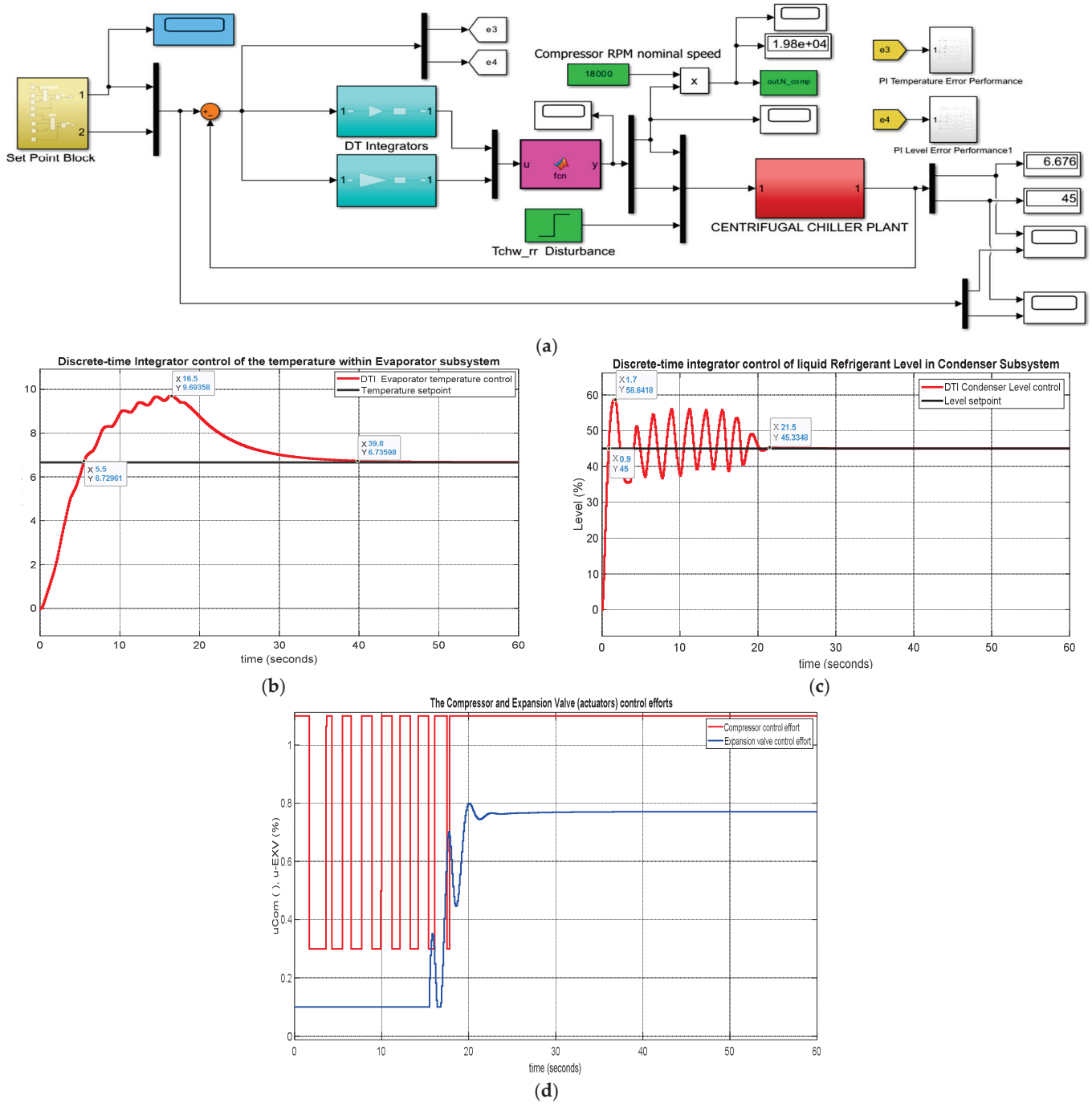
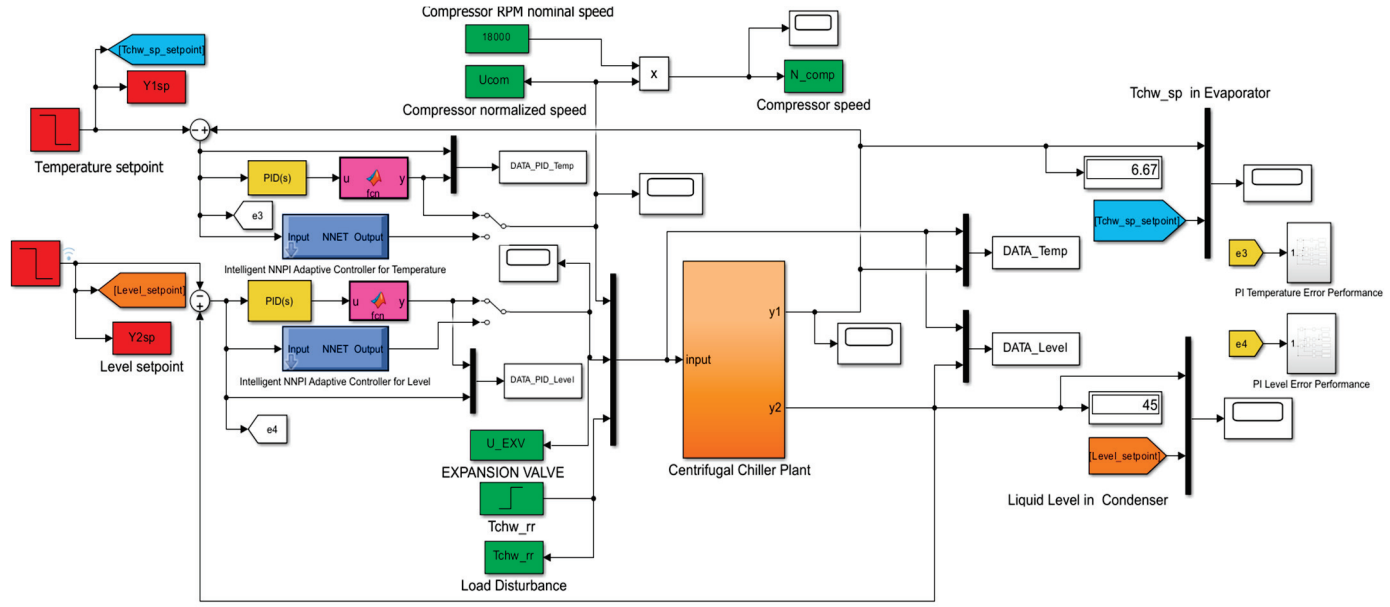
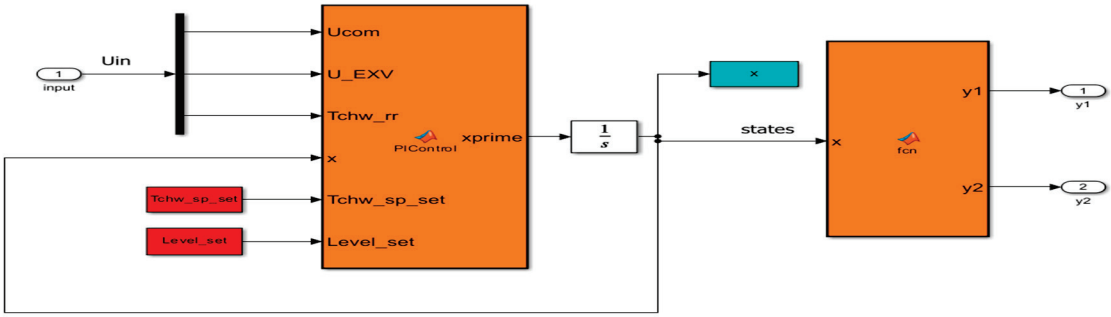


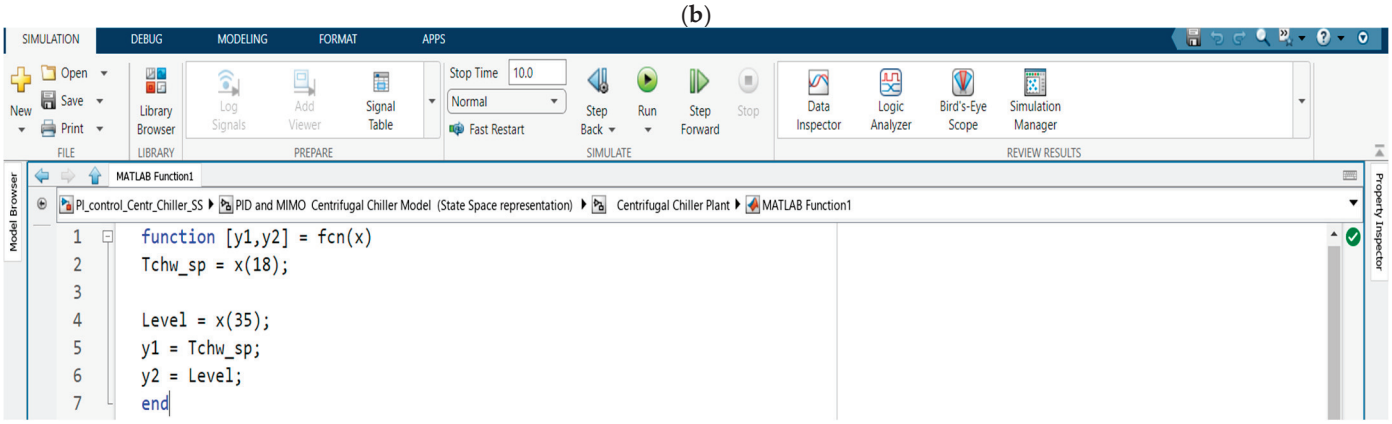
Figure A2. DTI control—MIMO Centrifugal Chiller system extended model and Simulink simulations: (**a**) DTI control Simulink diagram; (**b**) chilled water temperature in evaporator; (**c**) liquid refrigerant level in condenser; (**d**) compressor and expansion valve-opening actuator control efforts.



(**a**)

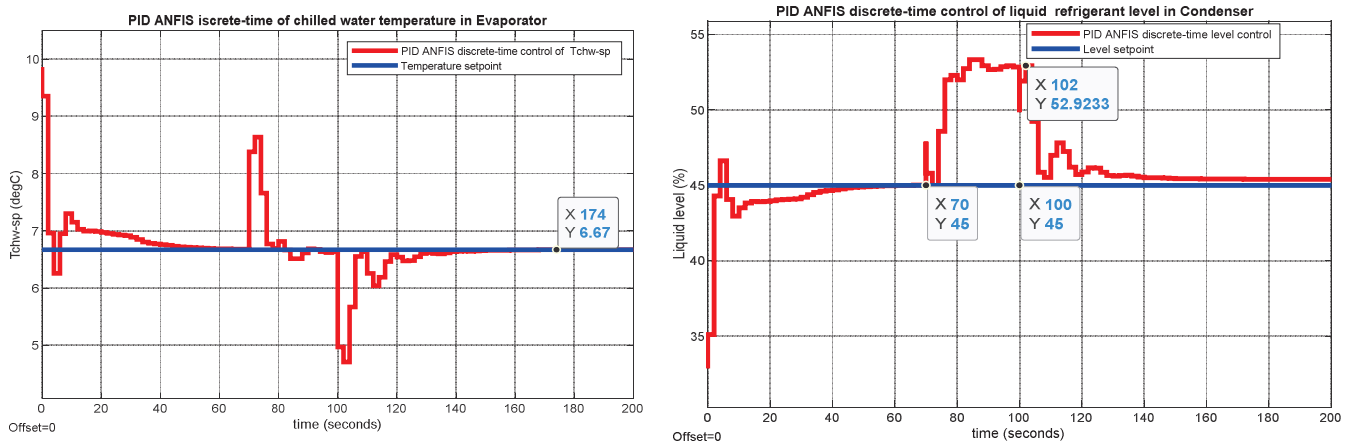


(**b**)



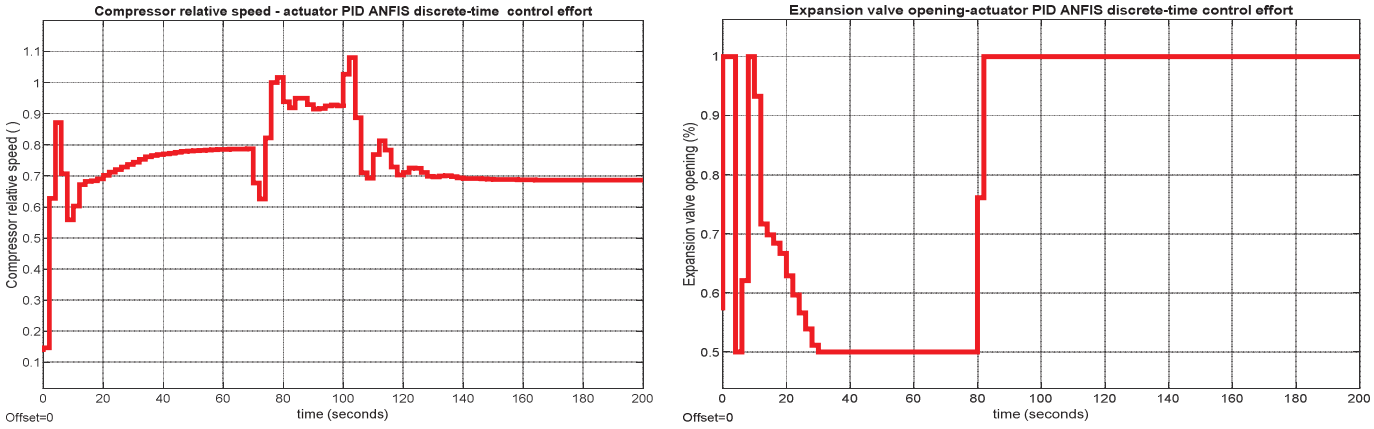
(**c**)

Figure A3. Simulink model of PID closed-loop control strategies for the MIMO Centrifugal Chiller nonlinear extended model (39 states): (**a**) overall diagram; (**b**) components of the Simulink block diagram (MATLAB function of the extended model and MATLAB function visualization block); (**c**) MATLAB function visualization block.



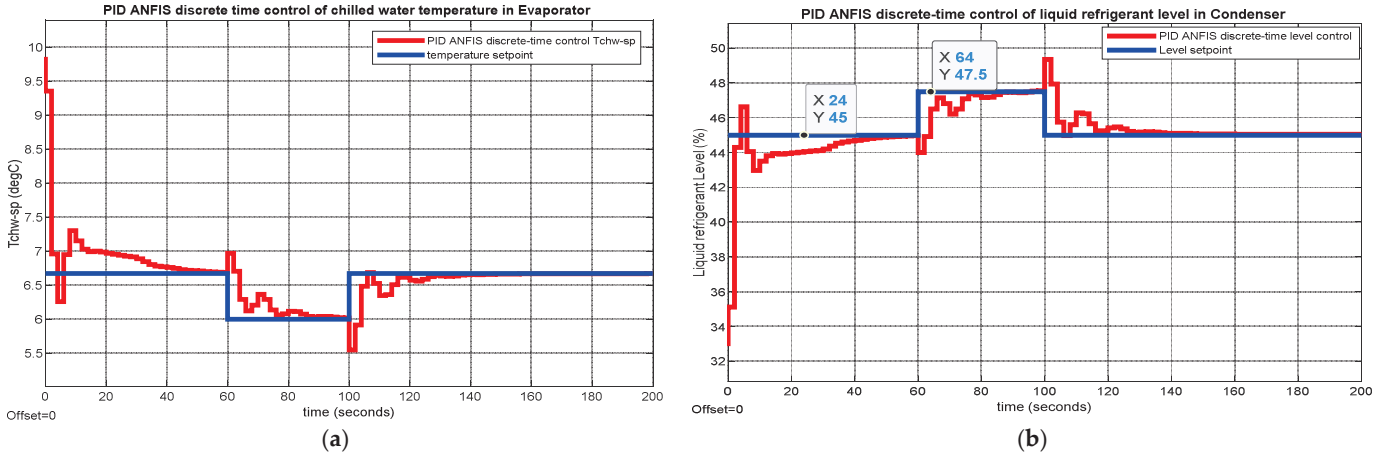
(**a**) (**b**)

Figure A4. PID ANFIS discrete-time simulation results, rejection temperature disturbance Tchw_rr: (**a**) chilled water temperature inside evaporator; (**b**) liquid refrigerant level within condenser.



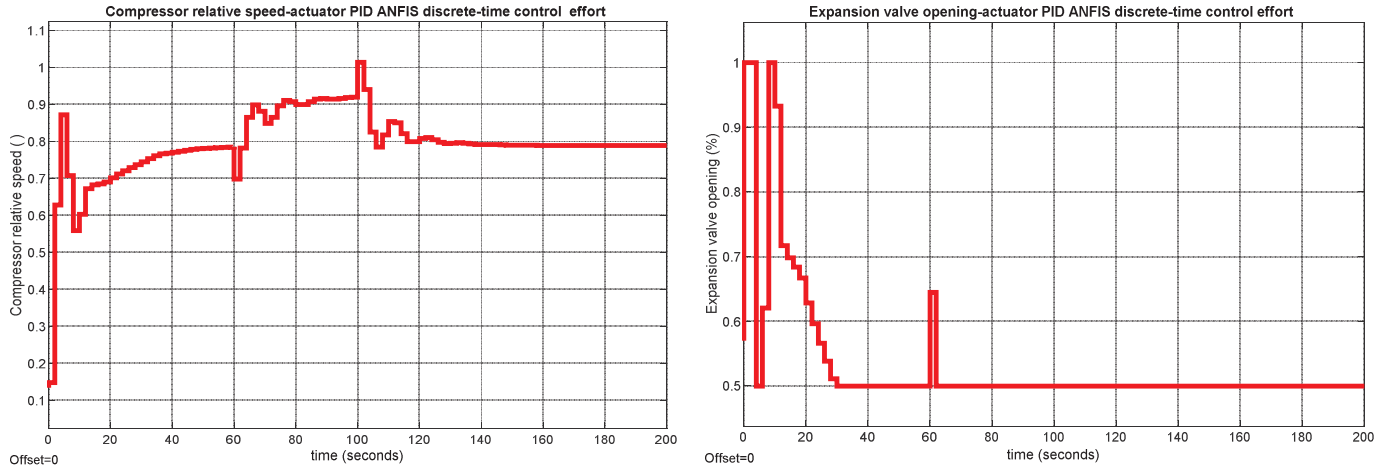
(**a**) (**b**)

Figure A5. PID ANFIS discrete-time control MATLAB Simulink simulation results of actuator control effort: (**a**) compressor relative speed; (**b**) expansion valve opening.



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

Figure A6. PID ANFIS discrete-time control MATLAB Simulink simulation results: (**a**) chilled water temperature within evaporator; (**b**) liquid refrigerant level inside condenser.



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

Figure A7. PID ANFIS discrete-time control—MATLAB Simulink simulation results of actuator control effort: (**a**) compressor relative speed; (**b**) expansion valve opening.

Figure 3: Data Statistics

Data Statistics for: MPC Temperature control

Select statistics to display on the figure:

	X	Y
min	0	0
max	60	6.772
mean	30	6.277
median	30	6.67
mode	0	0
std	17.36	1.189
range	60	6.772

Help | Save to Workspace...

(**a**)

Figure 3: Data Statistics

Data Statistics for: RL DLNN MPC Temperature c...

Select statistics to display on the figure:

	X	Y
min	0	0
max	60	6.76
mean	30	6.261
median	30	6.752
mode	0	0
std	17.36	1.222
range	60	6.76

Help | Save to Workspace...

(**b**)

Figure 4: Data Statistics

Data Statistics for: MPC Level control

Select statistics to display on the figure:

	X	Y
min	0	0
max	60	45.01
mean	30	44.4
median	30	45
mode	0	0
std	17.36	3.655
range	60	45.01

Help | Save to Workspace...

(**c**)

Figure 4: Data Statistics

Data Statistics for: RL DLNN level control

Select statistics to display on the figure:

	X	Y
min	0	0
max	60	46.75
mean	30	45.84
median	30	46.36
mode	0	0
std	17.36	4.042
range	60	46.75

Help | Save to Workspace...

(**d**)

Figure A8. Statistical data tables: (**a**) MPC evaporator temperature control; (**b**) RL DLNN evaporator temperature control; (**c**) MPC condenser refrigerant level control; (**d**) RL DLNN MPC condenser refrigerant level control.

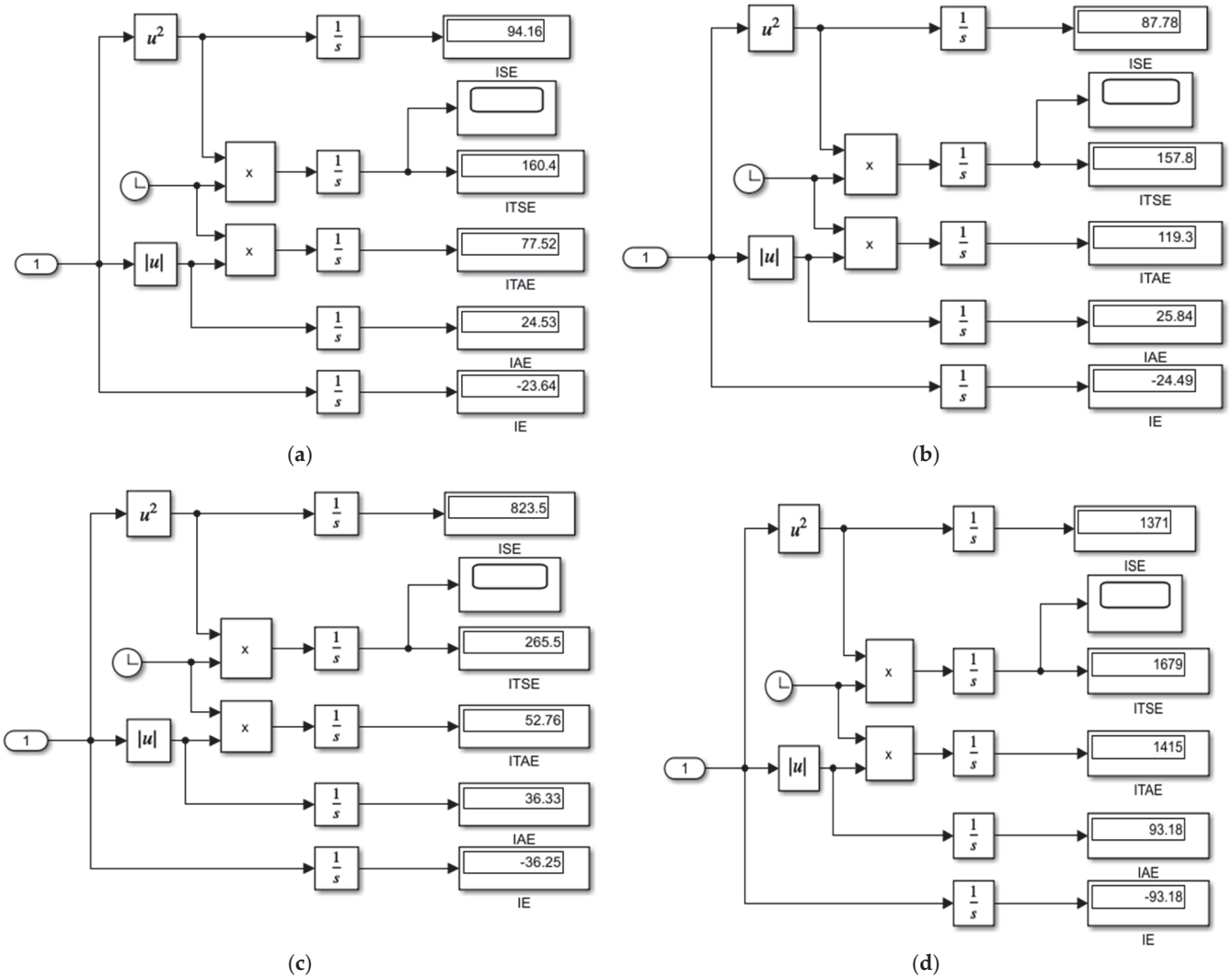


Figure A9. Performance error indicators: (**a**) MPC evaporator temperature control; (**b**) RL DLNN evaporator temperature control; (**c**) MPC condenser refrigerant level control; (**d**) RL DLNN MPC condenser refrigerant level control.

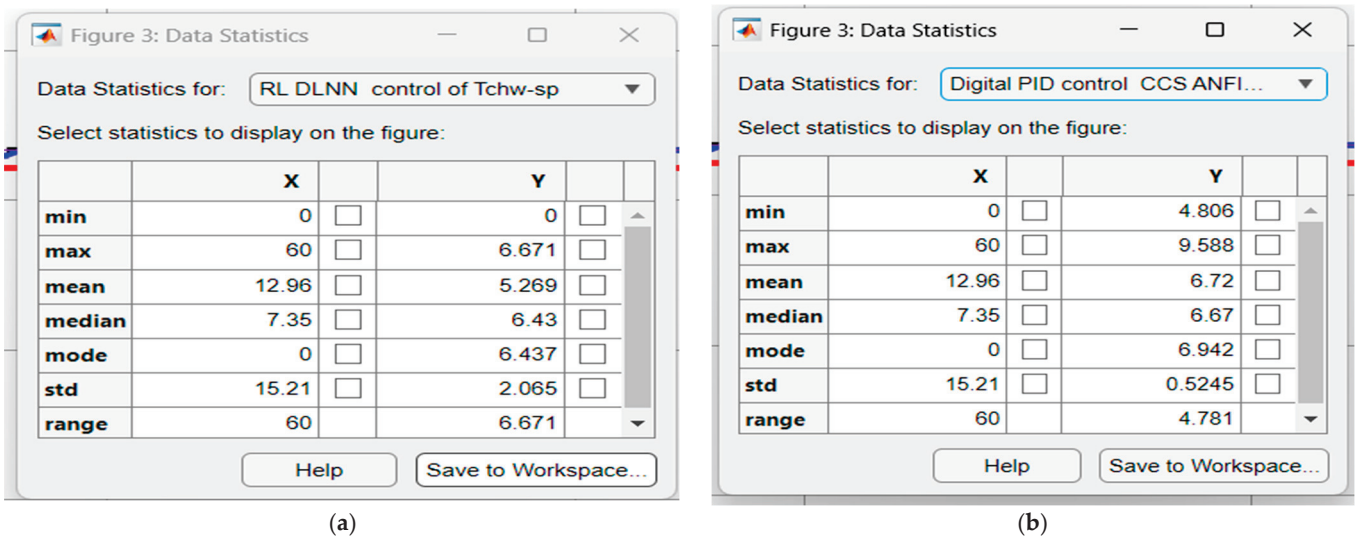
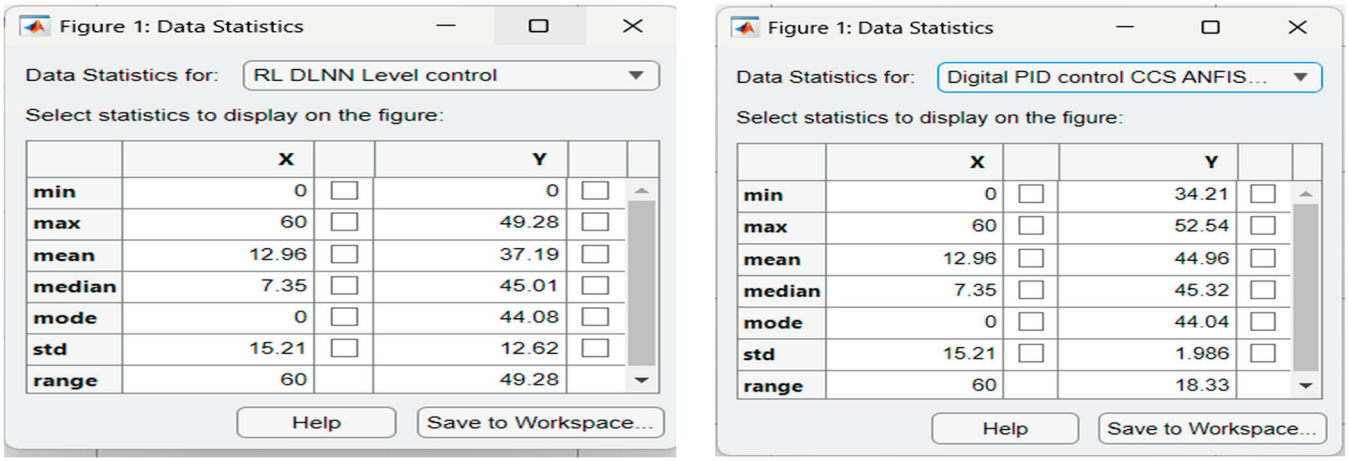
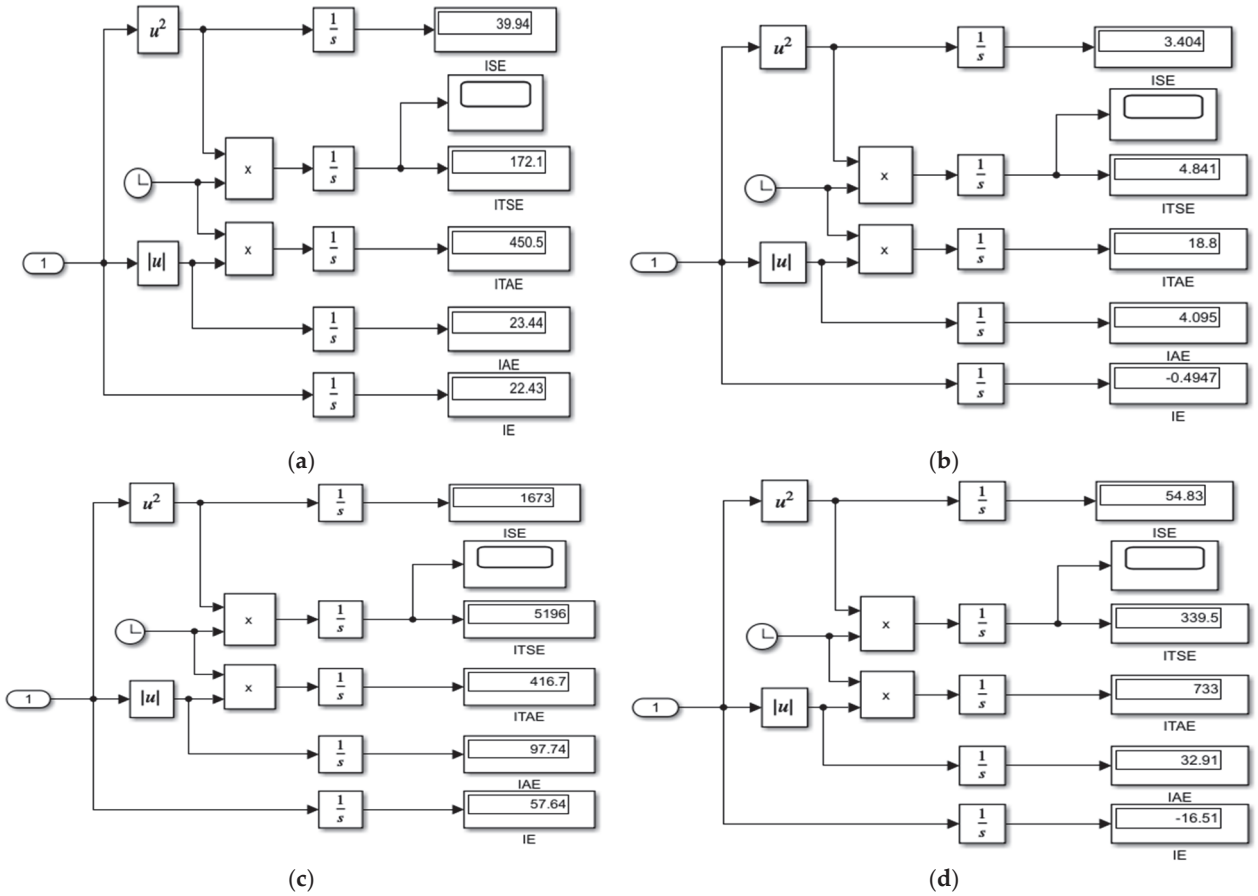


Figure A10. *Cont.*



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

Figure A10. Statistical data tables: (**a**) RL DLNN evaporator temperature control; (**b**) digital PID evaporator temperature control—CCS ANFIS model; (**c**) RL DLNN MPC condenser refrigerant level control; (**d**) digital PID condenser refrigerant level control—CCS ANFIS model.



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

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

Figure A11. Performance control error indicators: (**a**) RL DLNN evaporator temperature control; (**b**) digital PID evaporator temperature control—CCS ANFIS model; (**c**) RL DLNN condenser refrigerant level control; (**d**) digital PID condenser refrigerant level control—CCS ANFIS model.

Appendix B

Appendix B.1 Algorithms

Algorithm A1. Generate Reward Function Based on MPC Specifications for a MIMO Centrifugal Chiller System-MPC RL DLNN

Step 1. Plant Dynamics (State Space):

```
clear
clc
close all
A=[0 -0.7601 0 0; 1 1.756 0 0; 0 0 0 -0.6884;0 0 1 1.628];
B= [0 0.006884 0; -0.1128 -0.03254 0.004691; 0  1.04  0; 0.7347 -1.535 0.01728];
C=[0 0.25 0 0; 0 0 0 2];
D=[0 0 0;0 0 0];
Ts =0.1;
plant = ss(A,B,C,D,Ts);
```

Step 2: Create an MPC object:

```
plant = ss(A,B,C,D);
plant.InputName = {'Ucom','uEXV','Trr'};
plant.InputGroup.MV = 1:2;
plant.InputGroup.MD = 3;
plant.OutputName = {'Tchw','Level'};
```

Step 2.1. Specify the sampling time (Ts), prediction horizon (p) and control horizon (m):

```
Ts = 0.1;
p = 10;
m = 3;
```

Step 2.2. Create the MPC object

```
mpcobj = mpc(plant,Ts,p,m);
```

Step 2.3. Open the Simulink model of the plant and MPC controller:

```
mdl = "rlMPC_CentrChiller_modified";
open_system(mdl);
```

Step 2.4. Specify the minimum and maximum values for manipulated variables (MVs):

```
mpcobj.MV(1).Min = 0;
mpcobj.MV(1).Max = 1.1;
mpcobj.MV(2).Min = 0;
mpcobj.MV(2).Max = 1;
```

Step 2.5. Nominate precise values for outputs:

```
mpcobj.Model.Nominal.Y = [6.67 45];
```

Step 2.6. Nominate precise values for all inputs:

```
mpcobj.Model.Nominal.U = [1.1 1 48];
```

Step 2.6. Specify output tuning weights:

```
mpcobj.Weights.OV = [10 10];
```

Step 2.7. Define the MPC object using structure format for output variables, tuning weights, and manipulated output variables:

```
mpcobj = mpc(plant, 0.1, 10, 2, mpcobj.Weights,mpcobj.MV,mpcobj.OV);
```

Step 3. Generate the reward function:

```
generateRewardFunction(mpc1obj)
type rewardFunctionmpcplant
```

Algorithm A1. *Cont.*

Step 3.1. Open the reward function in the Simulink model and append a new MATLAB function r = rewardFunctionMPC (y,refy,mv,refmv,lastmv):

```
open_system("rlMPC_CentrChiller_modified/Reward Function")
```

Step 4: Create an RL environment (Ts, Tsim, observations and actions specifications):

```
Tf = 60;
Ts = 0.1;
```

```
numObs = 24;
numAct = 2;
oinfo = rlNumericSpec([numObs 1]);
ainfo = rlNumericSpec([numAct 1], ...
    LowerLimit=[0 0]', ...
    UpperLimit=[1.1 1]');
```

```
blk = "rlMPC_CentrChiller_modified/RL Agent";
env = rlSimulinkEnv(mdl, blk, oinfo, ainfo);
```

Step 5: Create a RL agent (random seed for reproducibility, layer graph object—connect and define network path, plot criticNet):

```
rng(0)
```

```
mainPath = [
    featureInputLayer(numObs)
    fullyConnectedLayer(128)
    concatenationLayer(1, 2, Name="concat")
    reluLayer
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(1)];
actionPath = [
    featureInputLayer(numAct)
    fullyConnectedLayer(8, Name="fc_act")];

criticNet = layerGraph(mainPath);
criticNet = addLayers(criticNet, actionPath);
criticNet = connectLayers(criticNet, "fc_act", "concat/in2");
```

```
plot(criticNet);
```

Step 6. Define actorNet, plot it and create a deterministic actor function:

```
actorNet = [
    featureInputLayer(numObs)
    fullyConnectedLayer(128)
    reluLayer
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(numAct)];
```

```
plot(layerGraph(actorNet));
```

```
actordlNet = dlnetwork(actorNet);
actor = rlContinuousDeterministicActor(actordlNet, oinfo, ainfo);
```

Step 7. Specify the agent, critic optimizer, and actor optimizer options:

```
agentOpts = rlTD3AgentOptions(SampleTime=Ts, ...
    DiscountFactor=0.995, ...
    ExperienceBufferLength=1e6, ...
    MiniBatchSize=256);
```

```
for idx = 1:2
    agentOpts.CriticOptimizerOptions(idx).LearnRate = 1e-3;
    agentOpts.CriticOptimizerOptions(idx).GradientThreshold = 1;
end

agentOpts.ActorOptimizerOptions.LearnRate = 1e-3;
agentOpts.ActorOptimizerOptions.GradientThreshold = 1;
```

```
agentOpts.ExplorationModel.StandardDeviationMin = 1e-3;
agentOpts.ExplorationModel.StandardDeviation = 100;
agentOpts.ExplorationModel.StandardDeviationDecayRate = 1e-5;
```

Algorithm A1. *Cont.*

Step 8. Create the TD3 agent:

```
agent = rlTD3Agent(actor, [critic1,critic2], agentOpts);
validateEnvironment(env)
```

Step 9. Train the TD3 agent (options, evaluation, random seed, training function):

```
trainOpts = rlTrainingOptions(...
    MaxEpisodes=1000, ...
    MaxStepsPerEpisode=ceil(Tf/Ts), ...
    StopTrainingCriteria="EvaluationStatistic", ...
    StopTrainingValue=-0.2, ...
    ScoreAveragingWindowLength=20);
```

```
evl = rlEvaluator( ...
    NumEpisodes=1, ...
    EvaluationFrequency=50);
```

```
rng(0,"twister");
```

```
%doTraining = false;
doTraining=true;
if doTraining
    trainResult = train(agent, env, trainOpts, Evaluator=evl);
else
    load('rlDCServomotorTD3Agent.mat')
end
```

Step 10. Validate the closed-loop controller:

```
sim(mdl);
```

Algorithm A2. Generate Reward Function from a Step Response Block for a MIMO Centrifugal Chiller System—RL DLNN

Step 1. MIMO ARMAX Centrifugal Chiller model dynamics (state space):

```
clc
clear
close all
A=[0 -0.7601 0 0; 1 1.756 0 0; 0 0 0 -0.6884;0 0 1 1.628];
B= [0 0.006884 0; -0.1128 -0.03254 0.004691; 0  1.04  0; 0.7347 -1.535 0.01728];
C=[0 0.25 0 0; 0 0 0 2];
D=[0 0 0;0 0 0];
```

Step 1.1 Initialize the model:

```
Level0=0;
Levelf=60;
Tchw_sp0=0;
Tchw_spf=12;
Tf =20;
Ts =0.1;
```

Step 1.2 Open the Simulink model:

```
mdl = "rlCentrChiller_MIMO_StepResp_TwoAgs_SS";
open_system(mdl)
```

Step2. Open BLk1 and BLk2 step response blocks:

```
verifBlk1 = mdl + "/TempStepResponse";
open_system(verifBlk1)
verifBlk2 = mdl + "/LevelStepResponse";
open_system(verifBlk2)
```

Algorithm A2. *Cont.*

Step 3. Generate the reward functions Vfb1 and Vfb2:

```
generateRewardFunction(verifBlk1)
type rewardFunctionVfb1.m
generateRewardFunction(verifBlk2)
type rewardFunctionVfb2.m
```

Step 4. Combine observations and actions specifications:

```
oinfo = {oinfo1,oinfo2};
ainfo = {ainfo1,ainfo2};
```

Step 5. Create an RL environment:

```
env = rlSimulinkEnv(mdl, agentBlk, oinfo, ainfo);
```

Step 6. Create a Simulink environment interface:

```
agentBlk = mdl + ["/RL Agent1", "/RL Agent2"];
```

Step 7. Create a learning environment:

```
env = rlSimulinkEnv(mdl, agentBlk, oinfo, ainfo);
```

Step 8. Create the reinforcement learning environment for RL Agent1 and RL Agent2 (TDR3s):

Step 8.1. Set the random reproducibility seed:

```
rng(0,"twister")
```

Step 8.2. Define network path:

```
mainPath = [
    featureInputLayer(numObs)
    fullyConnectedLayer(128)
    concatenationLayer(1, 2, Name="concat")
    reluLayer()
    fullyConnectedLayer(128)
    reluLayer()
    fullyConnectedLayer(1)];
actionPath = [
    featureInputLayer(numAct)
    fullyConnectedLayer(8, Name="fc_act")];
```

Step 8.3. Create a layer graph object for criticNet:

```
criticNet = layerGraph(mainPath);
criticNet = addLayers(criticNet, actionPath);
```

Step 8.4. Connect all network layers:

```
criticNet = connectLayers(criticNet, "fc_act", "concat/in2");
```

Step 9. Plot the critic network structure:

Step 9.1. Convert network to dlnetwork:

```
criticdlNet = dlnetwork(criticNet, Initialize=false);
```

Step 9.2. Convert the critic functions for TD3 agents:

```
critic1 = rlQValueFunction(initialize(criticdlNet), oinfo1, ainfo1);
critic2 = rlQValueFunction(initialize(criticdlNet), oinfo2, ainfo2);
```

Step 9.3. Define actorNet:

```
actorNet = [
    featureInputLayer(numObs)
    fullyConnectedLayer(128)
    reluLayer()
    fullyConnectedLayer(128)
    reluLayer()
    fullyConnectedLayer(numAct)];
```

Algorithm A2. *Cont.*

Step 9.4. Plot the actorNet:

```
plot(layerGraph(actorNet));
```

Step 9.5. Create a deterministic actor function:

```
actordlNet = dlnetwork(actorNet);
actor1 = rlContinuousDeterministicActor(actordlNet, oinfo1, ainfo1);
actor2 = rlContinuousDeterministicActor(actordlNet, oinfo2, ainfo2);
```

Step 9.6. Specify the agent options:

```
agentOpts = rlTD3AgentOptions( ...
    SampleTime=Ts, ...
    DiscountFactor=0.99, ...
    ExperienceBufferLength=1e6, ...
    NumWarmStartSteps=1024, ...
    NumEpoch=20, ...
    MiniBatchSize=256);
```

```
agentOpts.ExplorationModel.StandardDeviation = 0.5;
agentOpts.ExplorationModel.StandardDeviationDecayRate = 1e-5;
agentOpts.ExplorationModel.StandardDeviationMin = 0;
% agentOpts.ExplorationModel.StandardDeviation = 0.5;
agentOpts.ActorOptimizerOptions.LearnRate = 1e-3;
agentOpts.ActorOptimizerOptions.GradientThreshold = 1;
agentOpts.CriticOptimizerOptions(1).LearnRate = 1e-3;
agentOpts.CriticOptimizerOptions(1).GradientThreshold = 1;
agentOpts.CriticOptimizerOptions(2).LearnRate = 1e-3;
agentOpts.CriticOptimizerOptions(2).GradientThreshold = 1;
```

Step 10. Create the TD3 agents:

```
agent1 = rlTD3Agent(actor1, [critic1, critic2], agentOpts);
agent2 = rlTD3Agent(actor2, [critic1, critic2], agentOpts);
agents={agent1 agent2};
```

Step 11. Train the TD3 agents

```
trainOpts = rlMultiAgentTrainingOptions(...
    AgentGroups={1,2},...
    LearningStrategy=("decentralized"),...
    MaxEpisodes=200,...
   MaxStepsPerEpisode=ceil(Tf/Ts),...
   StopTrainingCriteria="AverageReward",...
   StopTrainingValue=[1,1],...
   ScoreAveragingWindowLength=20);
```

```
% doTraining = false;
doTraining =true;
if doTraining
trainingStats = train([agent1,agent2], env, trainOpts);
else
    load("rlTD3Agent.mat")
end
```

Step 12. Closed-loop simulation:

```
maxsteps=ceil(Tf/Ts);
simOpts = rlSimulationOptions(MaxSteps=maxsteps);
experience = sim(env,[agent1,agent2],simOpts);
set(0, 'GraphicsTimeout', Inf);
sim(mdl);
```

Appendix B.2

Appendix B.2.1 DLNN Table for Reinforcement Learning Based on MPC Specifications

Table A1. Simulation Parameters: Reinforcement Learning Actor Critic and actorNet for MPC specifications.

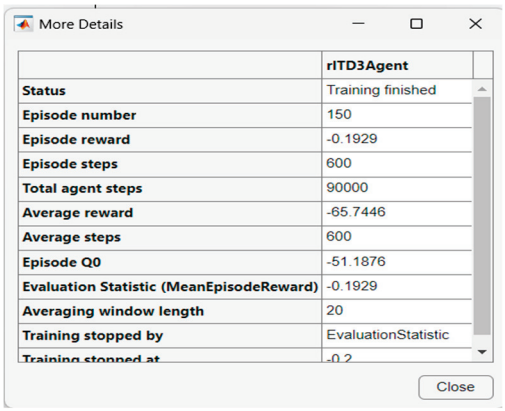
Parameter Name	Value/Description
RL Algorithm	TD3 RL agent **Description:** TD3 agents parametrize Q-value function **NN architecture:** NN with two inputs (one for observation and the second one for action-see the layer graphs for both actors) to model the parametrized Q-value function within both critics **Metrics:** • cumulative long-term reward (expected) • Average reward: −65.7446 at epoch 150 when the training process is finished after the agent reaches the stop training criteria (when evaluation statistic is −0.2); • Episode reward: −0.1929; • Episode Q0: −51.1876; • Evaluation statistic: −0.1929. * See the episode information from RL Training Monitor: 
DNN Architecture	1. **DLNN for actorCritic:** One Input layer of size 24 (number of observations), one input layer of size 2 (number of actions), three fully connected layers on the main path of output sizes 128, 64, and 1; One concatenation layer (1,2) and two RELU layers; One fully connected layer of size 8 on the action path. * See the layer graphs for Actor Critic and Actornet 2. **DLNN for actorNet:** One input layer of size 24 (number of observations or features); Three fully connected layers of sizes 128, 64, and 2 (number of actions); Two RELU layers
Reward Function	Derived from MPC model verification block
Training episodes	1000

Table A1. *Cont.*

Parameter Name	Value/Description
Test Episodes	1
Maximum Steps per Episode	600 (Tsim/Ts = 60/0.1)
Stop training criteria	Evaluation statistic: -0.2
Score Averaging Window length	20
Number Episodes for Evaluator	1
Evaluation frequency	50 episodes
Discount factor (γ)	0.995
Exploration model property	Noise: std = 100, exponential decay rate = 1×10^{-5}, minimum value reached of 1×10^{-3}.
Learning rate	0.001
Batch size	256
Replay Buffer Size	1×10^{6}

Appendix B.2.2 DLNN Table for Reinforcement Learning Based on Step Responses Specifications

Table A2. Simulation Parameters: Reinforcement Learning Actor Critic and actorNet for step responses specifications.

Parameter Name	Value/Description
RL Algorithm	Two TD3 agents for RL Agent1 (evaporator temperature) and RL Agent2 (condenser level). **Description:** two parametrized Q-value function approximators to estimate the value of the policy (expected cumulative long-term reward as metric). **NN architecture:** two NNs with two inputs for each agent (observation and action-see the layer graphs for Actor Critic and actorNet) that model the parametrized Q-value function within both critics. A critic function object is created to encapsulate the critic by wrapping around the critic deep neural network. To make sure the critics have different initial weights, each network is initialized before using them to create the critics. **Metrics:** • Cumulative long-term reward (expected metric); • Average reward: [0–50] at epoch 200 when the training process ends after the maximum number of episodes, even if both RL agents do not reach the training stop criteria (when average reward is set for both RL agents [1 1]); • Episode reward: [0–50]; • Episode Q0: [-1.493065–457.5267]. * See the episode information from RL Training Monitor.

	rlTD3Agent	rlTD3Agent_1
Status	Training finished	Training finished
Episode number	200	200
Episode reward	0	-50
Episode steps	1	1
Total agent steps	31045	31045
Average reward	0	-50
Average steps	1	1
Episode Q0	-1.493065	-457.5257
Averaging window length	20	20
Training stopped by	MaxEpisodes	MaxEpisodes
Training stopped at	Episode 200	Episode 200

Table A2. *Cont.*

Parameter Name	Value/Description
DNN Architecture	**DLNN for each Actor Critic:** One input layer of size 6 (number of observations), one input layer of size 1 (number of actions), three fully connected layers on the main path with output sizes 128, 64, and 1; One concatenation layer (1,2) and two RELU layers; One fully connected layer of size 8 on the action path. * See the layer graphs for Actor Critic and actorNet. **DLNN for actorNet:** One input layer of size 6 (number of observations or features); Three fully connected layers of sizes 128, 64, and 1 (number of actions); Two RELU layers.
Reward Function	Two reward functions that derive from step responses of two model verification blocks, one for evaporator temperature, other one for condenser level blocks.
Training episodes	200
Test Episodes	1
Maximum Steps per Episode	200 (Tsim/Ts = 20/0.1)
Stop training criteria	Average Reward: [1 1]
Score Averaging Window length	20
Number Episodes for Evaluator	1
Evaluation frequency	10 episodes
Discount factor (γ)	0.995
Exploration model property	Noise: std = 100, exponential decay rate = 1×10^{-5}, minimum decay rate value reached = 1×10^{-3}.
Learning rate	0.001
Batch size	256
Replay Buffer Size	1×10^{6}

References

1. Li, P.; Li, Y.; Seem, J.E. Modelica Based Dynamic Modeling of Water-Cooled Centrifugal Chillers. In Proceedings of the International Refrigeration and Air Conditioning Conference, West Lafayette, IN, USA, 12–15 July 2010; Purdue University: West Lafayette, IN, USA, 2010; pp. 1–8. Available online: https://docs.lib.purdue.edu/iracc/1091 (accessed on 10 June 2024).
2. Popovic, P.; Shapiro, H.N. Modeling Study of a Centrifugal Compressor. *ASHRAE Trans.* **1998**, *104 Pt 2*, 121.
3. Ning, M. Neural Network Based Optimal Control of HVAC&R Systems. Ph.D. Thesis, Department of Building, Civil and Environmental Engineering, Concordia University, Montreal, QC, USA, 2008.
4. Wong, S.P.W.; Wang, S.K. System Simulation of the performance of a Centrifugal chiller Using a Shell-and-Tube-Type Water-Cooled Condenser and R-11 as Refrigerant. *ASHRAE Trans.* **1989**, *95*, 445–454.
5. Wang, S.; Wang, J.; Burnet, J. Mechanistic Model of a Centrifugal Chiller to Study HVAC Dynamics. *Build. Serv. Eng. Res. Technol.* **2000**, *21*, 73–83. [CrossRef]
6. Browne, M.W.; Bansal, P.K. Steady-State Model of Centrifugal Liquid Chillers. *Int. J. Refrig.* **1998**, *21*, 343–358. [CrossRef]
7. Li, S.; Zaheeruddin, M. A Model and Multi-Mode Control of a Centrifugal Chiller System: A Computer Simulation Study. *Int. J. Air-Cond. Refrig.* **2019**, *27*, 1950031. [CrossRef]

8. Tudoroiu, R.E.; Zaheeruddin, M.; Radu, S.M.; Budescu, D.D.; Tudoroiu, N. The Implementation and the Design of a Hybrid digital PI Control Strategy Based on MISO Adaptive Neural Network Fuzzy Inference System Models–A MIMO Centrifugal Chiller Case Study. In *Machine Learning Paradigms. Learning and Analytics in Intelligent Systems*; Tsihrintzis, G., Jain, L., Eds.; Springer: Cham, Switzerland, 2020; Volume 18. [CrossRef]
9. Zadeh, L.A. Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems-Selected Papers by Lotfi Zadeh. In *Advances in Fuzzy Systems—Applications and Theory*; Klir, G.J., Yuan, B., Eds.; State University of New York at Binghamton, USA: Binghamton, NY, USA, 1996; Volume 6, 840p. [CrossRef]
10. Haykin, S. *Neural Networks and Learning Machines*, 3rd ed.; Horton, M.J., Mars, D., Opaluch, W., Disanno, S., Eds.; Pearson Education, Inc.: Upper Saddle River, NJ, USA, 2009.
11. Zurada, J.M. *Introduction to Artificial Neural Systems*, 1st ed.; Davis, G., Quadrata, P.L., Eds.; West Publishing Company: St. Paul, MN, USA, 1992.
12. Ljung, L. System Identification Toolbox Getting Started Guide. The MathWorks, Inc.,1 Apple Hill Drive Natick, MA 01760-2098 © COPYRIGHT 1988–2023 by The MathWorks, Inc., MATLAB R2023a. Available online: https://control.dii.unisi.it/sysid/book/ident_gs.pdf (accessed on 12 June 2024).
13. Walia, N.; Singh, H.; Sharma, A. ANFIS: Adaptive Neuro-Fuzzy Inference System-A Survey. *Int. J. Comput. Appl.* **2015**, *123*, 32–38. [CrossRef]
14. Jang, R. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [CrossRef]
15. Almabrok, A.; Psarakis, M.; Dounis, A. Fast Tuning of the PID Controller in An HVAC System Using the Big Bang–Big Crunch Algorithm and FPGA Technology. *Algorithms* **2018**, *11*, 146. [CrossRef]
16. MathWorks Support Documentation-Help Center. Estimate ARMAX or ARMA Model. Available online: https://www.mathworks.com/help/ident/ref/armax.html (accessed on 12 June 2024).
17. Namdari, A.; Samani, M.A.; Durrani, T.S. Lithium-Ion Battery Prognostics through Reinforcement Learning Based on Entropy Measures. *Algorithms* **2022**, *15*, 393. [CrossRef]
18. MathWorks Support Documentation-Help Center. Reinforcement Learning Using Deep Neural Networks. Available online: https://www.mathworks.com/help/deeplearning/ug/reinforcement-learning-using-deep-neural-networks.html (accessed on 10 September 2024).
19. MathWorks Support Documentation-Help Center. Reinforcement Learning for Control Systems Applications. Available online: https://www.mathworks.com/help/reinforcement-learning/ug/reinforcement-learning-for-control-systems-applications.html (accessed on 10 September 2024).
20. MathWorks Support Documentation-Help Center. Generate Reward Function from a Model Predictive Controller for a Servomotor. Available online: https://www.mathworks.com/help/reinforcement-learning/ug/generate-reward-fcn-from-mpc-for-servomotor.html (accessed on 10 September 2024).
21. MathWorks Support Documentation-Help Center. Generate Reward Function from a Model Verification Block for a Water Tank System. Available online: https://www.mathworks.com/help/reinforcement-learning/ug/generate-reward-fcn-from-verification-block-for-watertank.html (accessed on 20 September 2024).
22. Uzunovic, T.; Zunic, E.; Badnjevic, A.; Miokovic, I.; Konjicija, S. Implementation of digital PID controller. In Proceedings of the IEEE 33rd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 24–28 May 2010; Volume 33, pp. 1357–1361.
23. Grosse, G.; Maddison, C.; Bae, J.; Pitis, S.; Introduction to Machine Learning. Lecture 11-Reinforcement Learning, University of Toronto, Fall. 2020. Available online: https://www.cs.toronto.edu/~rgrosse/courses/csc311_f20/slides/lec11.pdf (accessed on 23 February 2025).
24. Abdel-Jaber, H.; Devassy, D.; Al Salam, A.; Hidaytallah, L.; El-Amir, M. A Review of Deep Learning Algorithms and Their Applications in Healthcare. *Algorithms* **2022**, *15*, 71. [CrossRef]
25. Kranthi Kumar, P.; Detroja, K.P. Design of Reinforcement Learning based PI controller for nonlinear Multivariable System. In Proceedings of the 2023 European Control Conference (ECC), Bucharest, Romania, 13–16 June 2023; pp. 1–6. [CrossRef]
26. You, W.; Yang, G.; Chu, J.; Ju, C. Deep reinforcement learning-based proportional–integral control for dual-active-bridge converter. *Neural Comput. Appl.* **2023**, *35*, 17953–17953. [CrossRef]
27. MathWorks Support Documentation-Help Center. Resolving Low-Level Graphics Issues. Available online: https://www.mathworks.com/help/matlab/creating-plots/resolving-low-level-graphics-issues.html (accessed on 20 September 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Nonparametric Probability Density Function Estimation Using the Padé Approximation

Hamid Reza Aghamiri [1], S. Abolfazl Hosseini [2], James R. Green [1] and B. John Oommen [3,*,†]

[1] Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada; hamidrezaaghamiri@cmail.carleton.ca (H.R.A.); jrgreen@sce.carleton.ca (J.R.G.)
[2] Department of Electrical Engineering, Research Center for Developing Advanced Technologies of Electrical and Electronics Industry, Yadegar-e-Imam Khomeini (RAH) Shahre Rey Branch, Islamic Azad University, Tehran 1474833163, Iran; abolfazl.hosseini@modares.ac.ir
[3] School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada
* Correspondence: oommen@scs.carleton.ca
† B.J.O. was an Adjunct Professor with the University of Agder, Grimstad, Norway, while the work was being done.

Abstract

Estimating the Probability Density Function (PDF) of observed data is crucial as a problem in its own right, and also for diverse engineering applications. This paper utilizes two powerful mathematical tools, the concept of moments and the relatively little-known Padé approximation to achieve this. On the one hand, moments encapsulate crucial information that is central to both the "time-" and "frequency-"domain representations of the data. On the other hand, the Padé approximation provides an effective means of obtaining a convergent series from the data. In this paper, we invoke these established tools to estimate the PDF. As far as we know, the theoretical results that we have proven, and the experimental results that confirm them, are novel and rather pioneering. The method we propose is *nonparametric*. It leverages the concept of using the moments of the sample data—drawn from the unknown PDF that we aim to estimate—to reconstruct the original PDF. This is achieved through the application of the Padé approximation. Apart from the theoretical analysis, we have also experimentally evaluated the validity and efficiency of our scheme. The Padé approximation is asymmetric. The most unique facet of our work is that we have utilized this asymmetry to our advantage by working with two mirrored versions of the data to obtain two different versions of the PDF. We have then effectively "superimposed" them to yield the final composite PDF. We are not aware of any other research that utilizes such a composite strategy, in any signal processing domain. To evaluate the performance of the proposed method, we have employed synthetic samples obtained from various well-known distributions, including mixture densities. The accuracy of the proposed method has also been compared with that gleaned by several State-Of-The-Art (SOTA) approaches. The results that we have obtained underscore the robustness and effectiveness of our method, particularly in scenarios where the sample sizes are considerably reduced. Thus, this research confirms how the SOTA of estimating nonparametric PDFs can be enhanced by the Padé approximation, offering notable advantages over existing methods in terms of accuracy when faced with limited data.

Keywords: Padé approximation; moments; nonparametric probability estimation

1. Introduction

Most natural events are stochastic [1] because they involve uncertain interactions of random internal "environments". To study these events, researchers have applied probability theory, which is a tool encompassing a branch of mathematics that aids in the analysis of the behavior of stochastic events. Probability theory, in particular, and stochastic processes, in general, are among the most fundamental fields of mathematics. They have applications in signal processing, pattern recognition, classification, forecasting, medical diagnosis, and many other applications. In practice, probability distributions find application in diverse fields such as electron emission, radar detection, weather prediction, remote sensing, economics, noise modelling, etc. [2–5]. Once the probabilistic model is well-defined, subsequent analysis often involves the theory of "estimation" (synonymously, learning, or training, depending on the application domain). Such estimation is always valuable, but it is nonetheless challenging [6], especially when the available data are limited [7,8].

In this context, the PDF of a random variable is central to various aspects of the analysis and characterization of the data and the model. Many studies have emphasized the need for estimating the PDF [9], which is challenging due to the absence of efficient construction methods.

Suppose that X_N is a set of observed data with N samples randomly distributed with an unknown PDF. The given X_N is characterized by its PDF ($f(x)$) of the underlying random variable, which satisfies the fundamental properties of a PDF:

$$\begin{aligned} \int_{-\infty}^{+\infty} f(x)\,dx &= 1, \\ f(x) &\geq 0. \end{aligned} \tag{1}$$

Once the PDF has been estimated based on a well-defined criterion, one also needs an assessment method to compare $f(x)$ with the set of observed data to identify the accuracy of our estimation. This PDF can be used to study the statistical properties of the dataset, estimate future outcomes, generate more samples, and so on. Numerous methods have been proposed for estimating the PDF, and these can be classified into two approaches, namely, using parametric or nonparametric models

Unlike parametric models, which make assumptions about the parametric form of the PDF, nonparametric density estimation does not assume the form of the distribution from which the samples are drawn [10]. In parametric models, one develops a model to fit a probability distribution for the observed data, followed by the determination of the parameters of this distribution. The procedure is summarized into four steps:

1. Select some known candidate probability distribution models based on the physical characteristics of the process and known characteristics of the distribution of the available samples.
2. Determine the parameters of the candidate models.
3. Assess the goodness-of-fit of the fitted generative model using established statistical tests and graphical analysis.
4. Choose the model that yields the best results [2,11,12].

However, these methods cannot cover all data because some datasets do not follow specific distributional shapes. As a result, these methods may fail to accurately approximate the true but unknown PDF [3]. Nevertheless, they contribute to solving many applications.

In nonparametric density estimation, we resort to techniques to find a model to fit the idealistic probability distribution of the observed data [13]. For example, a histogram approximation is one of the earliest density estimation methods, introduced by Karl Pearson [14]. Here, the entire range of values is divided into a series of intervals, referred to as

"bins", followed by adding up the number of samples falling into each bin. Although the so-called bin width is an arbitrary value, extensive research has been done to find a robust way to indicate the effective bin width [15,16]. Moreover, some bins will be empty, while others may have few occurrences. Thus, the PDF will ultimately vary between adjacent bins because of fluctuations in the number of occurrences. By increasing the bin size, each bin tends to contain more samples [17]; however, many important details will be filtered out. This weakness causes the inferred PDF to be discontinuous, and so it cannot be viewed as an accurate method, especially if the derivatives of the histogram are required.

These drawbacks led to the development of more advanced methods, such as Kernel Density Estimation (KDE), Adaptive KDE, Gaussian Mixture Model (GMM), Quantized Histogram-Based Smoothing (QHBS), and the one-point Padé approximation model [18].

KDE [19] is a common density estimator resulting in smooth and continuous densities. It estimates the PDF by deploying a kernel function at each data point and then computing the sum of these kernels to construct a nonparametric estimate of the PDF. The kernel is usually a symmetric, differentiable function, such as the Gaussian. The bandwidth hyperparameter, which defines the width of the kernel, is crucial in determining the smoothness of the resulting estimate.

Adaptive KDE [20] is a form of KDE in which the bandwidth is not constant but instead changes according to the local density of the data. This approach allows for a more accurate depiction of the features of the distribution, especially in regions with dense data, while producing a smoother estimate where there are fewer data points.

GMM [21,22] is a parametric density estimation technique that assumes that the data have been produced by a weighted sum of Gaussian distributions. GMM models the mixture density by estimating the parameters of the (assumed) source Gaussian distributions. Each component of the mixture in the form of Gaussian has its own mean and covariance, while the overall density function is the weighted sum of these individual Gaussian densities. GMM offers a number of advantages over other models and can capture more complex and multi-modal distributions. It is, however, a parametric approach whose accuracy depends on the degree to which its underlying assumptions are met.

QHBS [23] is an approach to address the challenges posed by conventional histograms. It approximates the PDF by smoothing a histogram in a manner that reduces the discontinuities and enhances the smoothness of the resulting density estimate. This approach splits the data into sets of bins and then employs a quantization procedure to soften the estimates and thus produce a smoother, continuous, and differentiable representation of the PDF.

Each of these methods has its strengths and challenges. For instance, while KDE and Adaptive KDE offer smooth and continuous density estimates, selecting the appropriate kernel function and bandwidth can be difficult. On the other hand, methods like GMM and QHBS can address multimodal and discontinuous distributions effectively, but GMM requires knowledge of the number of component distributions, while QHBS is sensitive to the number of bins used in the underlying histogram. Despite these challenges, these methods are widely used in practice due to their ability to provide accurate and flexible density estimates in various applications.

Although we have presented only a rather brief review of these schemes, it can serve as a backdrop to our current work. In this paper, we shall argue that, unlike the previous methods, the use of the *moments* of the data can serve to yield an even more effective method to estimate the PDF, which also quite naturally leads us to the Padé transform, explained below. *Note that we will synonymously refer to this phenomenon as the Padé approximation and the Padé Transform.* Amindavar and Ritcey first proposed the use of the Padé approximation for estimating PDFs [24]. We include Amindavar's original method in our quantitative evaluations below, here referred to as the "one-point Padé approximation model". As originally

formulated, this approach suffers from several challenges, which are mentioned in [24]. First, the Padé approximation method is only applicable to strictly positive data. Second, this method exhibits substantial distortion on the left side of the approximated PDF. Third, the original method did not demonstrate accurate PDF modelling when moments must be estimated directly from sample data. As detailed below, our proposed approach overcomes these challenges to arrive at a novel density estimation technique.

1.1. Organization of the Paper

In the next section, we briefly recall the basic concepts of the Padé approximation and moment estimation, both of which are used frequently in this research, and will be necessary for developing our proposed method. The subsequent section will illustrate the one-point Padé approximation method, which is an essential section to facilitate understanding of the concepts of our work. After that, the proposed method is introduced in reasonable depth. To demonstrate the efficiency of our work, we examine our work by invoking several criteria. Thereafter, the comparison between the proposed method and some of the mentioned methods will be presented. As shown in Section 5, the proposed method is a robust, accurate, and automatic way to estimate the PDF. It would be an efficient way to find an unknown PDF for many applications, especially if the sample size is small.

1.2. On the Experimental Results

It is prudent if we say a few sentences about the datasets, distributions, and experimental results that we have presented here. It is, of course, infeasible and impossible to survey and test all the techniques for estimating PDFs, and to further test them for all the possible distributions. Rather, we have compared our new scheme against a competitive *representative* method in the time domain. With regard to the frequency domain, we have compared our scheme against the Padé approximation, whose advantages are discussed in a subsequent section. Further, all the algorithms have been tested for four distributions with distinct properties and characteristics, namely Gaussian mixtures, the beta, the Gamma, and the exponential distribution.

2. Mathematical Background

In this section, we briefly present the mathematical foundations of the phenomena used here, namely that of Moments and of the Padé Transform in Section 2.1 and Section 2.2, respectively.

2.1. Moments

Moments are a set of statistical parameters used to describe a distribution [25]. They provide information about the statistical characteristics of a distribution, which is used in many applications. Let $f(x)$ be the PDF of the random variable x. Then, for the continuous function $f(x)$, the MGF at zero is defined as:

$$M_n = E(x^n) = \int_{-\infty}^{\infty} x^n f(x)\, dx. \tag{2}$$

In practical scenarios, we encounter two main challenges. Firstly, the PDF of the random variable x is often unknown. Secondly, access to samples is typically limited. In many cases, when estimating moments, we resort to the discrete form of finding the MGF. Assuming we have N samples, the nth moment can be estimated as follows:

$$M_n = \hat{E}(x^n) = \frac{1}{N} \sum_{i=1}^{N} x_i^n. \tag{3}$$

While the number of moments is infinite, we will focus on utilizing a subset of these moments. Later, we will leverage these moments to describe the PDF, and thereafter, to the Padé Transform.

2.2. The Theoretical Aspects of the Padé Transform

The Padé approximation (or Transform) is a significant mathematical tool developed by Henry Padé around 1890 [26]. It finds numerous applications in engineering [27], but in the interest of brevity these are not detailed here. The concept of the Padé approximation involves representing a function in a rational form of the corresponding Taylor series expansion. There are several compelling reasons to use the Padé approximation. Firstly, it can be applied to accelerate the convergence of series. More interestingly, even in cases where a series does not converge, the Padé approximation can be employed and yields a good approximation that does converge. Secondly, the Padé approximation represents a *rational* form of a function, simplifying the problem in many instances. In this regard, the Padé approximation is a valuable tool for approximating functions [28], which is precisely the arena in which we are operating. It is fascinating that the existence of this rational function is derived from Cauchy's representation of analytic functions [24], and there is an equivalence of the Cauchy integral with the Riemann integral.

$$h(u) = \frac{1}{2\pi i} \int_C \frac{h(u')}{u' - u} \, du'. \tag{4}$$

In this equation, C represents a contour in the complex plane, over which the integration is performed. This equation holds true for all analytic functions. When employing the Riemann integral, this equation transforms into a rational function, as shown below:

$$h(u) = \frac{1}{2\pi i} \sum_n \frac{h(u_n)}{u_n - u} \Delta u_n, \tag{5}$$

where n is an index that runs over a set of discrete points, u_n, in the complex plane. Each point, u_n, corresponds to a specific value in this set, and the sum $\sum_n$ represents the contribution of all these points to the value of the function $h(u)$. We can thus see that all analytic functions can be transformed into rational functions. Later, we will observe that this conversion is essential for the convergence of $f(x)$ and for finding the inverse Laplace transformation. But for the present, we assume that we are given a power series representing an analytic function $f(u)$:

$$f(u) = \sum_{i=0}^{\infty} c_i u^n. \tag{6}$$

Our goal is to find a rational function, $P^{(L/M)}(u)$, to represent the series in the form of the Padé approximation function:

$$P^{(L/M)}(u) = \frac{\sum_{n=0}^{L} a_n u^n}{\sum_{n=0}^{M} b_n u^n}, \tag{7}$$

where L and M are the maximum degrees of the denominator and numerator, respectively. Now, to find a_n and b_n, we equate the RHSs of Equations (6) and (7) as:

$$\sum_{i=0}^{\infty} a_i u^n = \frac{\sum_{n=0}^{L} a_n u^n}{\sum_{n=0}^{M} b_n u^n} = \sum_{i=0}^{L+M} c_i u^n + O(u^{L+M+1}). \tag{8}$$

For this equation, we need $L + M$ coefficients [22]. Since we are dealing with practical problems, we assume that we have a finite number of available coefficients, and ignore the others. More specifically, $O(u^{L+M+1})$ represents the remaining terms that we have ignored in Equation (8). To obtain coefficients, we equate the two sides as:

$$\left(\sum_{n=0}^{M} b_n u^n\right)\left(\sum_{n=0}^{L+M} c_i u^n\right) = \sum_{n=0}^{L} a_n u^n. \tag{9}$$

By matching the coefficients, this system of linear equations can be solved easily using the so-called Hankel matrix:

$$\begin{bmatrix} C_{(L-M+1)} & C_{(L+M+2)} & \cdots & C_L \\ \vdots & \vdots & \vdots & \vdots \\ C_{(L-M+K)} & C_{(L-M+K+1)} & \cdots & C_{(L+K-1)} \\ \vdots & \vdots & \vdots & \vdots \\ C_L & C_{(L+M)} & \cdots & C_{(L+M-1)} \end{bmatrix} \begin{bmatrix} b_M \\ \vdots \\ b_K \\ \vdots \\ b_1 \end{bmatrix} = - \begin{bmatrix} C_{(L+1)} \\ \vdots \\ C_{(L+K+1)} \\ \vdots \\ C_{(L+M)} \end{bmatrix} \tag{10}$$

These equations calculate the coefficients of the denominator of Equation (8). To find the coefficients of the numerator of Equation (8), we apply Equation (10) to Equation (9) to yield:

$$a_j = C_j + \sum_{i=1}^{\min(M,j)} b_i c_{(j-i)} \quad 0 \le i \le L \tag{11}$$

Therefore, by finding the coefficients, we can see how a Taylor series can be transformed into a rational function. In the next section, we demonstrate how we can leverage the advantages of the rational function for our work, namely to obtain the PDF of the samples by using the one-point Padé approximation and moments estimation.

2.3. Finding the PDF of Samples Using Moments Estimation Function

The one-point Padé approximation [24] was proposed for estimating the distribution of continuous functions, by using the MGF. However, it has some drawbacks. This method fails to provide accurate estimates for all the training samples. Specifically, it only estimates the PDF for positive random variables and is not applicable to negative ones, limiting its applicability to many PDFs. Additionally, the final estimation exhibits substantial distortion near zero (on the left side of the approximation). In this paper, our goal is to address these problems.

In this section, we explain how we can estimate the PDF with the aid of the biased approximation of the moments and the one-point Padé approximation. Indeed, through method refinement, the estimation noise can be reduced, leading to improved performance. To simplify the problem, we initially focus on positive random variables with a finite number of samples. Later, we explore how we can estimate the PDF for datasets comprising both positive and negative values. For the purpose of notation, let $\{S_i\}$ with N coefficients be a set of samples for which we aim to determine the PDF. Let:

$$S_i = \{s_1, \ldots, s_N\} \tag{12}$$

Let M_n denote the nth moment of S_i. Further, let $f(x)$ be the continuous PDF of the random variable X, and $F(u)$ be its Laplace transformation, defined as:

$$F(u) = \int_{x=0}^{\infty} f(x) e^{-ux}\, dx. \tag{13}$$

We now expand the exponential function by using its Taylor series. By doing this, and simplifying the equation, we get:

$$F(u) = \int_0^\infty f(x) \sum_{n=0}^{\infty} \frac{(-u)^n}{n!} x^n \, dx = \sum_{n=0}^{\infty} \frac{(-u)^n}{n!} \int_0^\infty f(x) x^n \, dx = \sum_{n=0}^{\infty} M_n \frac{(-u)^n}{n!}. \tag{14}$$

As discussed in Section 2.1, where we briefly introduced the concept of moment estimation, we can use this approach to estimate M_n. In the next section, we will demonstrate how we can apply the Padé approximation. The primary reason for using the Padé approximation is that it transforms the infinite series of $F(u)$ into a rational function, which provides a good approximation which also converges more effectively. The rational function is represented as $\frac{P_L}{Q_D}$, where L and D denote the degrees of the numerator and denominator, respectively. Additionally, converting to a rational function enables us to apply the inverse Laplace transform to find $f(u)$.

2.4. Implementation Considerations of the Padé Transform

The first question that we must resolve is that of determining how many poles and zeros (i.e., the values of L and D) are adequate to yield an accurate approximation. For tackling this issue, we follow the arguments proposed by [29]. For uniqueness and convergence, we limit ourselves to Padé approximations where the degree of the numerator is less than the degree of the denominator. Thereafter, we invoke the residue theorem to find the inverse Laplace transformation as:

$$P_D^L(u) = \frac{\sum_{n=0}^{L} a_n u^n}{1 + \sum_{n=1}^{D} b_n u^n} = \sum_{i=1}^{D} \frac{\lambda_i}{u + p_i}. \tag{15}$$

$$\mathcal{L}^{-1}(P_D^L(u)) = f(x) = \sum_{i=1}^{M} \lambda_i e^{-p_i x}, \quad p_i > 0. \tag{16}$$

In Equation (15), λ_i represents the residue associated with each term in the decomposition and p_i denotes the poles of the rational function, where each p_i corresponds to a specific root of the denominator. One major issue with this method is the distortion incurred in estimating the PDF, near zero. As previously discussed, the accuracy of the one-point Padé approximation method is compromised in the right tail; hence, the left tail of the approximation is more reliable and accurate. To illustrate this issue, consider the example of a Gaussian PDF:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{17}$$

where σ and μ are the standard deviation and mean, respectively, as illustrated in Figure 1, where $\sigma = 50$ and $\mu = 15$. More specifically, the figure was drawn using 300 samples generated from this distribution. Indeed, we have estimated the PDF and compared it to the ideal (true) one. In Figure 1, we compare the ideal PDF with the estimated one.

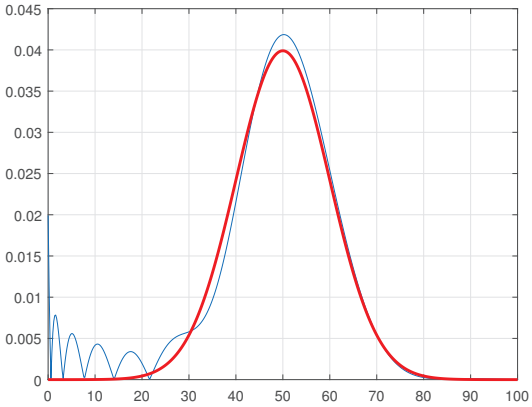


Figure 1. Comparison between the Gaussian function and its estimation using the Padé approximation. Here, the degree of the numerator and denominator are $L = 8$ and $D = 10$, respectively. The blue series represents the ideal PDF, while the red series represents the estimated PDF.

As seen in Figure 1, the left part of the estimated PDF does not resemble the shape of the original PDF. Therefore, we require a method to estimate the left part of the PDF accurately. In the next section, we will propose a solution to address this issue. Subsequently, we will compare the proposed method with other methods to assess its effectiveness.

3. The Padé Approximation Using *Artificial* Samples

Many branches of science have invoked the Padé approximation as a tool to convert a series into a rational function. Similarly, in our study, we have applied it indirectly to samples to approximate an unknown PDF. The previously discussed method estimated $f(x)$ using the Padé approximation. However, its accuracy decreases significantly as $x \to 0$, causing distortion in that portion of the PDF. Moreover, all the samples in the previous method must be positive. To resolve this, we propose to utilize two datasets. The first is the given dataset. Thereafter, we create a new dataset from the original one, and the two of them will simultaneously be utilized to solve the problem. Suppose a given sample is randomly distributed from an unknown distribution $f(x)$ (Equation (18)). We then generate N samples from R_n by mirroring R_n across the y-axis, creating the series L_n:

$$R_n = \{x_1, x_2, \ldots, x_N\}. \tag{18}$$

$$L_n = \{-x_1, -x_2, \ldots, -x_N\}. \tag{19}$$

As illustrated in Equation (19), all elements of the L_n series will be mirrored values of R_n. Consequently, the estimated PDF using the L_n dataset, denoted as $l(x)$, is precisely the reflection of our PDF function, $f(x)$, across the negative Y-axis. To address the weakness of the one-point Padé approximation, in the following we will estimate the left part of $f(x)$ using the right part $l(x)$. Consequently, we estimate the right part of $f(x)$ with the help of R_n, leading to a more accurate estimation of the function. Now, to calculate $l(x)$ from L_n, we follow the same procedure. To implement the one-sided Laplace transform, firstly, we need non-zero integer samples. Therefore, we shift all elements of L_n by the maximum amount of the L_n series + 1, which is $\beta + 1$:

$$L_n = \{-x_1 + \beta + 1, \ldots, -x_N + \beta + 1\}. \tag{20}$$

Proceeding now from Equation (3):

$$E((-x+\beta+1)^{n+1}) = \frac{1}{N}\sum_{n=0}^{N}(-x+\beta+1)^n \quad \hat{x} = -x+\beta+1 \quad \hat{M}_n = \frac{1}{N}\sum_{n=0}^{N}(\hat{x})^n. \tag{21}$$

$\hat{M}_n$ is the moment generating function of our new training samples. So we continue:

$$l(u) = \sum_{n=0}^{\infty} \hat{M}_n \left(\frac{(-u)^n}{n!} \right) \longrightarrow \hat{P}_D^L(u) = \frac{\sum_{n=0}^{L} a_n u^n}{1 + \sum_{n=1}^{D} b_n u^n} = \sum_{i=1}^{D} \frac{\hat{\lambda}_i}{u + \hat{p}_i}. \tag{22}$$

$$L^{-1}(\hat{P}_D^L(u)) = l(\hat{x}) = \sum_{i=1}^{M} \hat{\lambda}_i e^{-\hat{p}_i \hat{x}}, \quad \hat{p}_i > 0. \tag{23}$$

It is easy to see that $l(\hat{x})$ is the mirrored form of our desired PDF. However, due to the use of the Padé approximation, this estimation suffers from the same issue we discussed before: it is only reliable when $\hat{x}$ is sufficiently large. Now, our objective is to utilize the right side of $l(\hat{x})$ to replace the distorted part of $f(x)$. To achieve this, we need to roll back the entire shifting process. Considering Equation (19) and then combining Equation (21) and Equation (15), we have:

$$x = -\hat{x} + \beta + 1 \implies \hat{x} = -x + \beta + 1. \tag{24}$$

The estimated PDF $\hat{f}(x)$ is given by:

$$l(x) = \begin{cases} l(-x + \beta + 1), & \text{if } x \to 0 \\ r(x), & \text{if } x \to \infty. \end{cases} \tag{25}$$

$R(x)$ is the estimation of the series R_n. An acceptable point C is required to be considered as the "centroid" for combining these two functions. The final result is equal to:

$$g(x) = \begin{cases} \frac{l(x)}{\int_C^{\infty} \left(l(x) + \int_C^{\infty} r(-x+m)\, dx \right)}, & \text{if } x > C; \\ \frac{f(-x+m)}{\int_C^{\infty} \left(l(x) + \int_C^{\infty} f(-x+m)\, dx \right)}, & \text{if } x < C. \end{cases} \tag{26}$$

The denominators of sub-equations are necessary to ensure that the integral of $g(x)$ is equal to unity. Selecting the best value for C is one of the challenges that should be taken into account. In this paper, we select the mean of X_n because it yields good experimental results. For illustrative purposes, we apply the method in some examples. With this approach, one can approximate the function; however, a discontinuity may arise. The function is not smooth at the point C, and there is a jump at the point C. For illustrative purposes, we first apply the method in one example, and then we highlight the issue. Finally, we propose a method by which we can render the function to be continuous at this point. The ideal Gaussian PDF given in Equation (17) was used to generate normally (Gaussian) distributed samples. Thereafter, we randomly selected 200 samples, S_{200}, with the help of an inverse transform sampling method [30].

$$S_{200} = \{x_1, \ldots, x_{200}\}. \tag{27}$$

$$\begin{aligned} R_{200} &= \{x_1, \ldots, x_{200}\}, \\ L_{200} &= \{-x_i + \max(R_{200}) \mid 1 \le i \le 200\}. \end{aligned} \tag{28}$$

These two sets, namely R_{200} and L_{200}, are samples that are generated to calculate the right and left parts of the PDF, respectively. As shown in Figure 1, after applying the approximant method, the left and right parts of the PDF are calculated separately. For example, $\{L_{200}\}$ is the dataset utilized to draw the left part of the PDF.

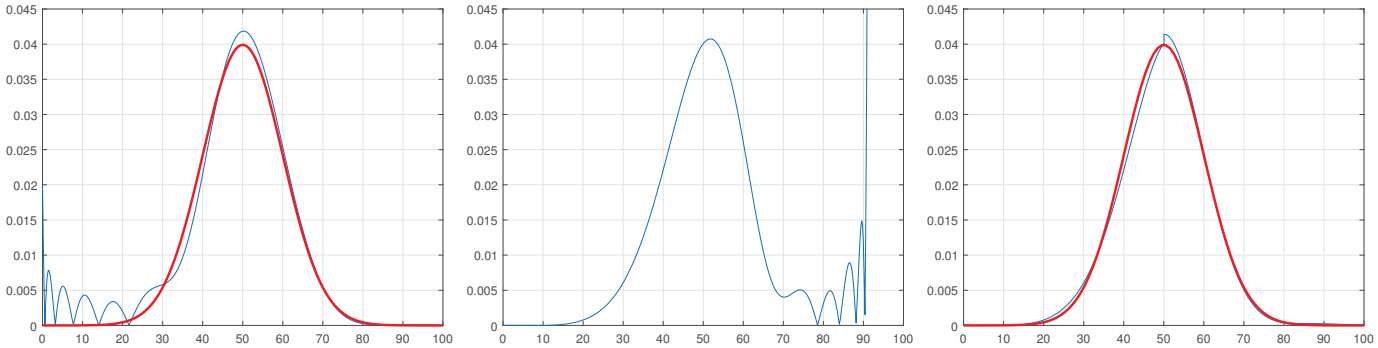
$$N(50, 10) = \frac{1}{\sqrt{2\pi \times 10}}\, e^{-\frac{(x-50)^2}{200}}. \tag{29}$$

In the same vein, $f_L(x)$ and $f_R(x)$ are functions that draw the left and right sides of the learned PDF, respectively. After performing this operation, we obtain:

$$\begin{aligned} f_L(x) &= 7200e^{-0.23x} - (255+71i)e^{-0.19-0.13ix} \\ &\quad + (255-71i)e^{-0.19+0.13ix} + (11+9.2i)e^{-0.15-0.19ix} \\ &\quad + (11-9.2i)e^{-0.15+0.19ix} + \ldots \end{aligned} \tag{30}$$

$$\begin{aligned} f_R(x) &= (88+31i)e^{-0.21-0.11ix} + (88-31i)e^{-0.21+0.11ix} \\ &\quad - (255+40i)e^{-0.22-0.056ix} - (255-40i)e^{-0.22+0.056ix} + \ldots \end{aligned} \tag{31}$$

In Figure 2, we display the left and right parts of the PDF. The blue series represents the ideal PDF, while the red series represents the estimated PDF. Subsequently, these PDFs are combined to create a less noisy PDF; see Figure 2c, which is highly similar to the ideal PDF. However, as we discussed earlier, there is a discontinuity at the midpoint (50). To address this, in the next section (Section 4) we introduce a sigmoid-based smoothing function to eliminate this discontinuity.



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

Figure 2. The results of estimating the Gaussian density function are presented. The blue line represents the estimated PDF, while the red line depicts the true source density function. As observed, the left tail of the estimated PDF (**a**) and the right tail of the estimated PDF (**b**) exhibit inaccuracies. The final result, shown in (**c**), is the fused estimate with a discontinuity where the left and right estimated PDFs join.

4. Using a Sigmoid-Based Transition

When dealing with piecewise functions, abrupt transitions between segments can introduce discontinuities or undesirable behaviour. To achieve a smooth transition between two functions, we can use a sigmoid-based blending approach [31]. This method is particularly useful in scenarios where continuity and differentiability are important, such as in working with PDFs or control systems. The sigmoid function is defined as:

$$\sigma(t,c,k) = \frac{1}{1+e^{-k(t-c)}}. \tag{32}$$

The sigmoid function is defined as a smooth, continuous curve that transitions from 0 to 1. Here, t is the independent variable, c is the transition point, and k controls the steepness of the transition. The parameter k determines how sharply the curve transitions. For small values of k, the transition is gradual, blending the functions smoothly over a wider range. For larger values of k, the transition becomes sharper, closely approximating a step function at $t = c$.

To smoothly blend two functions, $f_1(t)$ and $f_2(t)$, around the transition point, c, we define the blended function by Equation (33) as:

$$f_{\text{blended}}(t,c,k) = f_1(t) \cdot (1 - \sigma(t,c,k)) + f_2(t) \cdot \sigma(t,c,k). \tag{33}$$

This formulation ensures that as $t \to -\infty$, $\sigma(t,c,k) \to 0$, and the blended function approaches $f_1(t)$. Similarly, as $t \to \infty$, $\sigma(t,c,k) \to 1$, and the blended function approaches $f_2(t)$. Around $t = c$, the two functions are smoothly combined based on the sigmoid weighting. The sigmoid-based transition is commonly used in various applications, such as in control systems, where it facilitates gradual shifts between control regimes, and signal processing, where it is employed for blending signals or applying gradual filters. This approach provides a robust mechanism for creating smooth, continuous transitions between functions, enhancing both stability and interpretability in mathematical models. In our work, we used the sigmoid function for PDF reconstruction. This function ensures smooth transitions between the mirrored and original PDFs, effectively eliminating the discontinuity point.

The result of applying the smoothing algorithm to the example from Section 3 is seen in Figure 3.

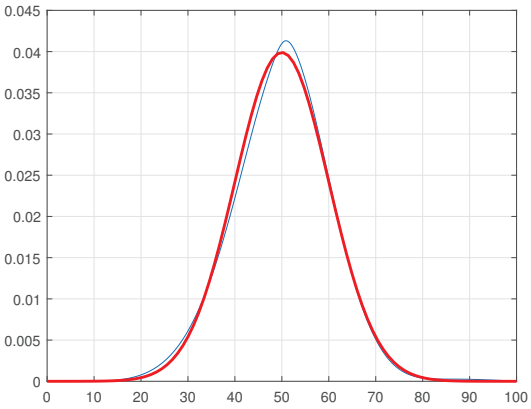


Figure 3. The Padé approximation of the Gaussian distribution function (blue line) compared with the true density function (red line) after applying the sigmoid smoothing algorithm.

To illustrate the proposed method within the broader context, Figure 4 presents a schematic diagram of the PA procedure. This figure outlines the sequential steps required to achieve an accurate approximation. Specifically, using Equation (33), we depict the final estimation obtained by combining the two estimations.

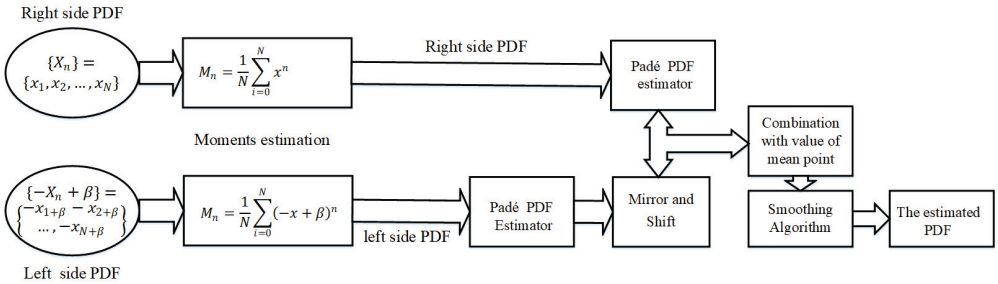


Figure 4. Schematic diagram of the PA procedure. After invoking the "merging" phase, one obtains the final estimation.

In addressing the challenge of estimating the probability density function for negative random variables, we utilized a methodology involving horizontally shifting all samples to create a new dataset comprising only positive values. This positive dataset was then

utilized to accurately approximate the probability density function. The process was initiated by aggregating all samples with a maximum absolute value, and subsequently, employing the same procedure outlined in Section 3 to estimate the PDF. In the final step, the samples were reverted to their original positions. To illustrate this procedure, consider the example in which S_{500} is a series of 500 samples randomly generated from a Gaussian distribution with a mean of zero and a variance of 20, i.e., $(N(0, 20))$.

In the first step, all the samples were transformed into positive samples by shifting by the $\text{Min}(S_{1000})$, such that they were strictly positive.

$$S_{500} = \{x_1, \ldots, x_{500}\}. \tag{34}$$

The transformed samples, denoted as S'_n, were obtained using the following equation:

$$S'_n = S_n + \text{Max}(S) + 1. \tag{35}$$

The proposed method was then applied to this set which results in an estimate of the PDF (see Figure 5).

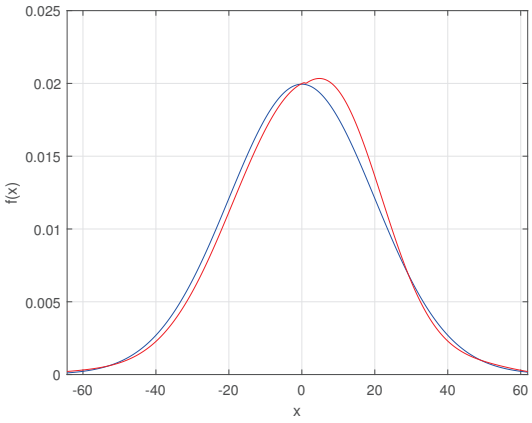


Figure 5. Estimation of Gaussian density function by considering the negative values. The blue and red lines show the estimated and ideal density functions, respectively.

5. Quantitative and Comparative Assessment

In this section, we report the results obtained by conducting several experiments to assess the performance of the proposed method. The experiments can be summarized as follows. Firstly, the proposed method was applied to datasets generated from well-known probability distributions. The results were then compared with the ideal PDF. Subsequently, the proposed method was compared with other state-of-the-art methods. We aimed to show that our new approach could produce favourable outcomes when working with fewer samples, i.e., in scenarios where data are limited, which is a common occurrence in real-world situations. In addition, we have made the MATLAB code, implementing all the proposed methods, publicly available in the following GitHub repository: https://github.com/hamidddds/Twosided_PadeAproximation (accessed on 23 January 2025).

In our research project, we conducted a series of tests to thoroughly investigate the problem using three datasets of varying sizes. The first dataset consisted of 300 samples, which we categorized as a *large* dataset, providing a robust representation for comprehensive analysis. The second dataset contained 100 samples, classified as a *medium-sized* dataset, offering a sufficient number of samples for a balanced evaluation. Lastly, the third dataset comprised only 50 samples, representing a *very small* dataset specifically designed to simulate challenging scenarios in which access to extensive data is limited. This enabled us to assess how well the performance varied across the sample sizes. In real-world appli-

cations, sample sizes of 50 are often insufficient to accurately calculate all data properties in specific ranges. In many cases, scientists address this limitation by making assumptions about the underlying structure of the density and then using parametric methods to estimate the parameters, although the overall accuracy of the results tends to decrease, particularly when incorrect assumptions are made regarding the density. Despite these challenges, we included these small sample sizes to demonstrate that, even in harsh conditions, our method can capture some characteristics of the data and perform reasonably well. However, it is important to note that such small sample sizes are highly sensitive to data sparsity, which can significantly impact performance. Clearly, it is important to mention that the accuracy of all estimations improved with increasing numbers of samples.

In this article, we employ five techniques for comparison with our proposed approach: Kernel Density Estimation (KDE), Adaptive KDE, Gaussian Mixture Model (GMM), Quantized Histogram-Based Smoothing (QHBS), and the one-point approximation model. Since our method is based on the one-point Padé approximation model, we have included it as a reference point for evaluating our approach. The KDE is a nonparametric technique that smooths data points to estimate the underlying PDF without assuming a specific distribution. It employs a kernel function, such as the Gaussian, and a bandwidth parameter to control the level of smoothing. In this study, a Gaussian kernel was used, with the bandwidth calculated using Silverman's rule of thumb [32]. The Adaptive KDE enhances the standard KDE by dynamically adjusting the bandwidth based on the density of data points. This adaptability enables it to handle varying data densities efficiently, capturing intricate details in densely populated regions while ensuring uniformity in sparser areas. The Adaptive KDE utilized the same Gaussian kernel but varied the bandwidth locally to reflect data density. The GMM is a parametric technique that represents the PDF as a combination of weighted Gaussian components. It estimates the parameters using methods such as expectation–maximization, providing an effective approach for clustering and density estimation. While the GMM performs well when data adhere to Gaussian assumptions, it may face challenges with non-Gaussian distributions. In this study, a three-component Gaussian model was employed, a configuration commonly observed in many applications [33–35]. The QHBS combines histogram-based density estimation with smoothing techniques. It divides the data into segments, or bins, estimates the density within each bin, and then applies smoothing to reduce abrupt changes in the data distribution. For this study, the number of bins was set to 20 to balance detail and smoothness [36].

To evaluate the performance of our proposed approach, we utilized six metrics: the **Wasserstein Distance**, the **Bhattacharyya Distance**, the **Correlation Coefficient**, the **Kullback–Leibler (KL) Divergence**, the **L1 Distance**, and the **L2 Distance**. The **Wasserstein Distance** evaluates the cost of transforming one probability distribution into another by examining variations in their distributions, making it responsive to the distribution's shape and useful for assessing alignment. A score of zero for the Wasserstein Distance signifies similarity between distributions. The **Bhattacharyya Distance** measures the overlap between two distributions, helping us understand how well two methods approximate densities by directly reflecting their probabilistic similarity. A smaller Bhattacharyya Distance suggests a better approximation quality, with zero indicating a perfect overlap. The **Correlation Coefficient** assesses the linear relationship between distributions, providing a clear interpretation of their alignment. A Correlation Coefficient of 1 signifies perfect linear correlation, whereas a value of 0 indicates no linear relationship between them. The **KL Divergence** measures the difference between two distributions, revealing the drawbacks of approximating one using the other. It is useful for identifying subtle

variations in the distribution of probability mass. A lower KL Divergence is preferred, as it signifies greater similarity between distributions, with zero indicating identical distributions. The **L1 Distance** (also known as Manhattan or Taxicab Distance) calculates the sum of absolute differences between corresponding elements in two vectors or distributions. It is particularly effective in high-dimensional spaces and provides an intuitive measure of dissimilarity. A smaller L1 Distance indicates greater similarity between distributions, with zero representing identical distributions. The **L2 Distance** (commonly referred to as Euclidean Distance) measures the straight-line distance between two points in a multidimensional space using the Pythagorean theorem. It is widely used for numerical data and provides a geometric interpretation of similarity. Like L1, an L2 Distance of zero indicates identical distributions, while larger values signify greater dissimilarity. These measures collectively provide a comprehensive foundation for assessing the accuracy, reliability, and alignment of density estimation methods across various dimensions and scenarios.

5.1. Example 1: Gaussian Mixture

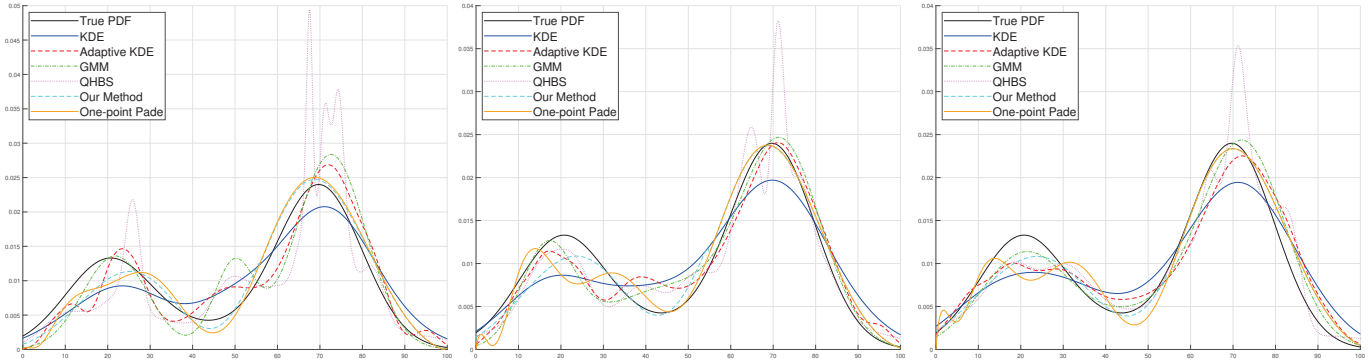
For the first experiment, we chose a Gaussian mixture distribution model. This is because many real-world problems are based on Gaussian distributions or their combinations so as to simulate real-world problems. The multi-modal distribution was constructed by summing three Gaussian distributions:

$$f(x) = \frac{1}{0.75}\left(\frac{1}{2}N(20,10) + N(40,20) + \frac{1}{4}N(70,10)\right) \tag{36}$$

The value of 0.75 ensured that the properties of a probability distribution were preserved, with its integral being equal to unity. In this equation, $N(\mu,\sigma)$ represents a Gaussian distribution with mean μ and standard deviation σ, as defined in Equation (37). The coefficients before the Gaussian distributions were chosen arbitrarily.

The results of the experiments are displayed in Figure 6.

$$N(\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{37}$$



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

Figure 6. Visual assessment of different density estimation methods. The true Gaussian mixture PDF (black), KDE (blue), Adaptive KDE (red), GMM approximation (green), QHBS (pink), One-point approximation (orange), and our method (cyan). The results for different sample sizes are shown from left to right: 50 samples (**a**), 100 samples (**b**), and 300 samples (**c**).

Tables 1–3 detail the outcomes for varying sample sizes. We observed that, for small and medium and large sample sizes, our method outperformed the strategies proposed in prior studies. As the sample size increased, the GMM technique surpassed our method in

performance. The GMM's reliance on the underlying distribution has been noted for its effectiveness in estimating Gaussian density functions. However, when handling datasets where the histogram shape diverges significantly from a Gaussian distribution, its accuracy tends to decline. This makes parameter estimation challenging for methods like the GMM. Our method excels in such scenarios, as evidenced by Examples 2, 3, and 4 below. Notably, it produced results without visible distortions at the beginning of the graph, highlighting its robustness under these conditions.

Table 1. Performance metrics for different methods for the Gaussian mixture using 300 samples. Here and elsewhere, bold is used to indicate the best results.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	32.4425	2.3137	0.9324	0.0045	0.2329	0.0084
Adaptive KDE	28.8019	2.3091	0.9369	0.0029	0.1939	0.0074
GMM	**28.2692**	**2.3050**	0.9672	**0.0013**	0.1312	0.0053
QHBS	30.1941	2.3082	0.9272	0.0026	0.1877	0.0087
One-Point Padé	30.4723	2.3088	0.9496	0.0032	0.1605	0.0065
Proposed Method	31.4479	2.3057	**0.9712**	0.0016	**0.1163**	**0.0049**

Table 2. Performance metrics for different methods for the Gaussian mixture using 100 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	30.5430	2.3143	0.9195	0.0048	0.2430	0.0087
Adaptive KDE	28.4085	2.3094	0.9494	0.0031	0.1782	0.0065
GMM	**28.2237**	2.3105	0.9501	0.0038	0.1590	0.0068
QHBS	28.9642	2.3102	0.9237	0.0034	0.2033	0.0092
One-Point Padé	28.7078	2.3088	0.9572	0.0036	0.1462	0.0061
Proposed Method	29.5758	**2.3049**	**0.9742**	**0.0013**	**0.1185**	**0.0047**

Table 3. Performance metrics for different methods for the Gaussian mixture using 50 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	39.6778	2.3137	0.9191	0.0046	0.2352	0.0083
Adaptive KDE	38.3835	2.3138	0.9243	0.0052	0.2331	0.0086
GMM	39.8234	2.3127	0.9452	0.0049	0.2189	0.0095
QHBS	40.0201	2.3255	0.8182	0.0095	0.3795	0.0170
One-Point Padé	40.4137	2.3102	0.9659	0.0045	0.1729	0.0062
Proposed Method	**38.8917**	**2.3074**	**0.9672**	**0.0025**	**0.1567**	**0.0059**

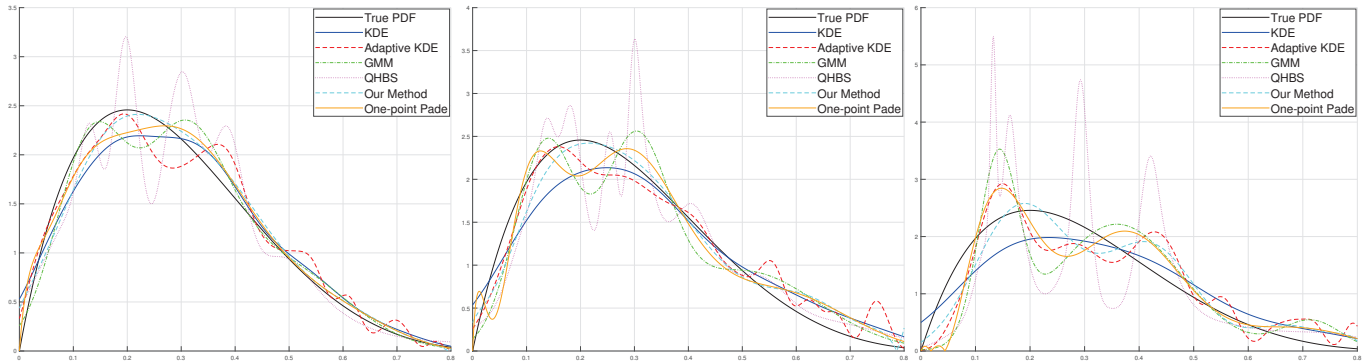
5.2. Example 2: The Beta Distribution

The next distribution model that we consider is the Beta distribution. The Beta distribution is a continuous probability distribution defined on the interval [0, 1], making it suitable for modelling probabilities and proportions. It is governed by two shape parameters, α and β, which influence the distribution's shape, allowing it to be symmetric, skewed, or U-shaped. The Beta distribution is extensively utilized in Bayesian statistics as a conjugate prior for binomial and Bernoulli processes, enabling efficient updates of beliefs based on observed data. In our experiment, the parameters were chosen as $\alpha = 2$ and $\beta = 5$ to evaluate the performance of our method in comparison to other methods.

The formula for the PDF of the Beta distribution is shown in Equation (38). The results of the simulations are given in Figure 7.

$$f(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \quad \text{for} \quad 0 \le x \le 1 \tag{38}$$

The results (see Tables 4–6) demonstrate that our method outperformed competing approaches and secured first place in the competition. For a sample size of 50, the results indicate that KDE achieved superior performance in terms of the Correlation Coefficient, showing an advantage in this specific criterion. However, across other evaluation metrics our method consistently delivered better performance.



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

Figure 7. Visual assessment of different density estimation methods. The true Beta distribution (black), KDE (blue), Adaptive KDE (red), GMM approximation (green), QHBS (pink), One-point approximation (orange), and our method (cyan). The results for different sample sizes are shown from left to right: 300 samples (**a**), 100 samples (**b**), and 50 samples (**c**).

Table 4. Performance metrics for different methods for the beta distribution using 300 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	18.4653	7.1327	0.9844	0.005	0.1070	0.0042
Adaptive KDE	15.7541	7.1331	0.9800	0.017	0.1203	0.0048
GMM	11.6768	7.1313	0.9868	0.009	0.0806	0.0035
QHBS	11.8088	7.1360	0.9356	0.003	0.1780	0.0081
One-Point Padé	13.7342	7.1319	0.9876	0.021	0.0798	0.0034
Proposed Method	**11.0687**	**7.1311**	**0.9897**	**0.003**	**0.0789**	**0.0032**

Table 5. Performance metrics for different methods for the Beta distribution using 100 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	29.5680	7.1375	0.9811	0.012	0.1418	0.0055
Adaptive KDE	33.4228	7.1376	0.9848	0.015	0.1460	0.0055
GMM	28.7827	7.1368	0.9565	0.020	0.1716	0.0063
QHBS	37.6501	7.1395	0.9107	0.025	0.2023	0.0092
One-Point Padé	28.6066	7.1376	0.9644	0.018	0.1503	0.0058
Proposed Method	**27.1665**	**7.1356**	**0.9856**	**0.010**	**0.1105**	**0.0045**

Table 6. Performance metrics for different methods for the Beta distribution using 50 samples.

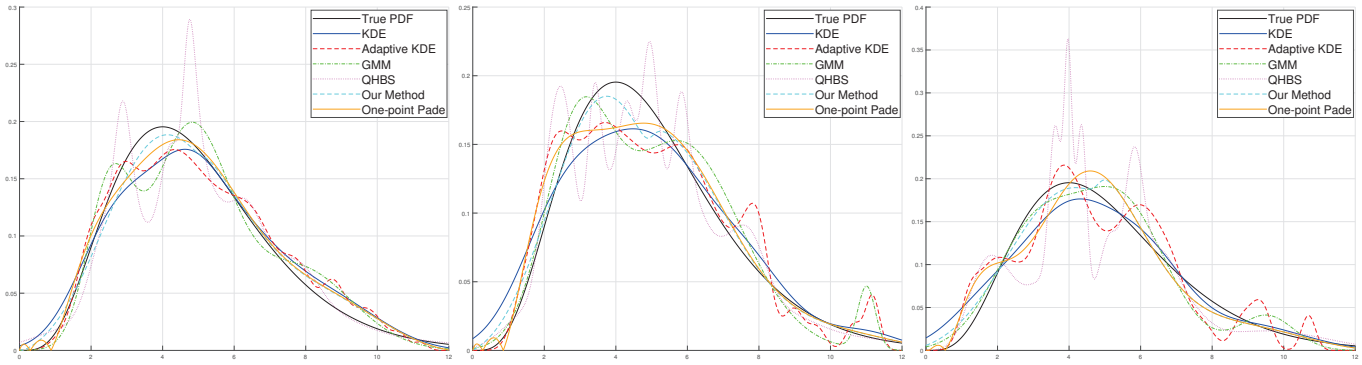
Method	↓ WD	↓ BD	↑ CC	↓ KLD	↓ L1	↓ L2
KDE	41.9417	7.1406	**0.9634**	0.025	0.2126	0.0074
Adaptive KDE	45.4668	7.1438	0.9130	0.030	0.2402	0.0088
GMM	43.3645	7.1566	0.8302	0.020	0.3128	0.0132
QHBS	51.2981	7.1775	0.6701	0.035	0.5102	0.0225
One-Point Padé	43.3572	7.1519	0.8915	0.018	0.2500	0.0100
Proposed Method	**37.9675**	**7.1405**	0.9392	**0.015**	**0.1921**	**0.0074**

5.3. Example 3: The Gamma Distribution

In our study, we also examined the Gamma distribution, which is recognized for its versatility in effectively representing both symmetrical and skewed datasets. This particular distribution provides valuable perspectives when estimating PDFs. It finds applications in fields such as finance (for predicting waiting durations), engineering (for analyzing failures), and various scientific phenomena that involve the duration until a specific event occurs. Noteworthy is the fact that the Exponential distribution, Erlang distribution, and chi-squared distribution are special cases of the Gamma distribution. The formula for the Gamma distribution is given in Equation (39).

$$f(x;\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \quad \text{for } x,\alpha,\beta > 0, \Gamma(\alpha) = (\alpha-1)! \tag{39}$$

In our experiment, we assumed that α was equal to 5 and β was equal to 1. The results, as shown in Figure 8 and Tables 7–9, demonstrate the effectiveness of the proposed method, where it outperforms all other methods on all performance metrics and for all sample sizes.



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

Figure 8. Visual assessment of different density estimation methods. The true Gamma PDF (black), KDE (blue), Adaptive KDE (red), GMM approximation (green), QHBS (pink), One-point approximation (orange), and our method (cyan). The results for different sample sizes are shown from left to right: 300 samples (**a**), 100 samples (**b**), and 50 samples (**c**).

Table 7. Comparison of PDF approximation methods for the Gamma distribution using 300 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	19.4452	4.4273	0.9888	0.00025	0.1274	0.0053
Adaptive KDE	15.8213	4.4260	0.9840	0.00027	0.1285	0.0051
GMM	14.5563	4.4266	0.9680	0.00033	0.1496	0.0065
QHBS	**9.7451**	4.4308	0.9265	0.00031	0.1876	0.0101
One-Point Padé	13.4854	4.4250	0.9926	0.00024	0.0899	0.0037
Proposed Method	13.9661	**4.4245**	**0.9950**	**0.00200**	**0.0826**	**0.0032**

Table 8. Performance metrics for different methods for the Gamma distribution using 100 Samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	18.6965	4.4309	0.9822	0.00265	0.1552	0.0065
Adaptive KDE	15.2305	4.4316	0.9506	0.00285	0.1996	0.0081
GMM	9.8272	4.4266	0.9519	0.00310	0.1474	0.0079
QHBS	8.0250	4.4290	0.9394	0.00360	0.1900	0.0090
One-Point Padé	9.4705	4.4255	0.9772	0.00235	0.1226	0.0057
Proposed Method	**7.3515**	**4.4245**	**0.9928**	**0.00205**	**0.0794**	**0.0035**

Table 9. Performance metrics for different methods for the Gamma distribution using 50 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	12.0644	4.4329	0.9810	0.0250	0.1355	0.0058
Adaptive KDE	12.8983	4.4406	0.9301	0.0300	0.2514	0.0096
GMM	11.0408	4.4308	0.9749	0.0275	0.1451	0.0058
QHBS	15.4045	4.4440	0.8379	0.0350	0.3498	0.0163
One-Point Padé	11.4033	4.4283	0.9657	0.0225	0.1563	0.0067
Proposed Method	**10.5327**	**4.4272**	**0.9872**	**0.0200**	**0.1052**	**0.0041**

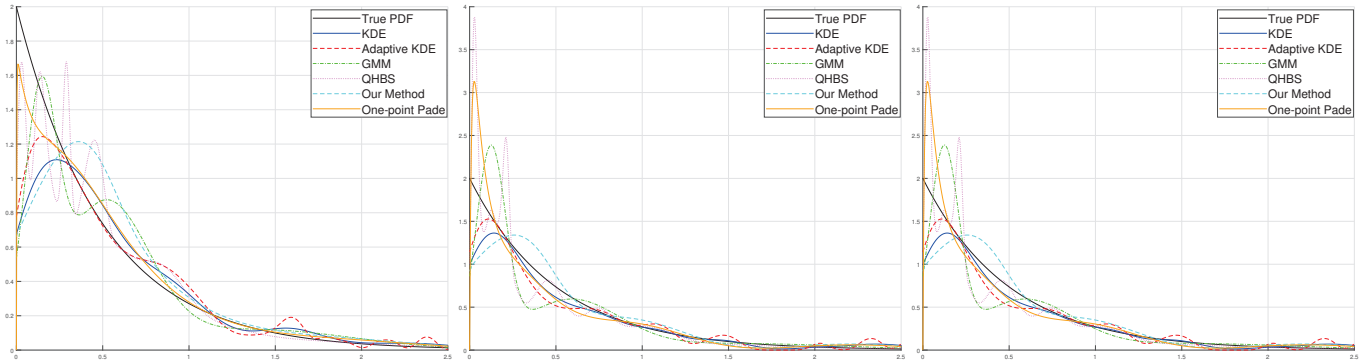
5.4. Example 4: The Exponential Distribution

The above example illustrates the nature of moment-based methods, which are flexible with a limited number of samples. The next distribution model that we consider is the Exponential distribution. The Exponential distribution is commonly used to model the time between events in a Poisson process and is defined by a rate parameter, λ. The formula for the Probability Density Function (PDF) of the Exponential distribution is shown in Equation (40). In our experiments, λ was set to 1. The results of the simulations are given in Figure 9.

$$f(x;\lambda) = \lambda e^{-\lambda x} \quad \text{for} \quad x \geq 0 \tag{40}$$

Upon reviewing the data shown in Figure 7 and Tables 10–12, it is noticeable that the effectiveness of all techniques has decreased. The main cause for this decline is attributed to the fact that the Exponential distribution presents difficulties in capturing its characteristics near zero, which poses a challenge for the techniques to accurately depict its traits in this region. In some instances, it seems that the One-Point Padé approximation demonstrates better performance. The reason for this could be that the One-Point Padé approximation introduces a distortion near zero, which in this scenario appears advantageous at first glance. The distortion aids in approximating the behaviour near zero to match the characteristics

of the Exponential function in that region. Nevertheless, it is crucial to realize that this can give a false impression of enhanced performance, as the distortion does not accurately represent the true nature of the Exponential distribution.



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

Figure 9. Visual assessment of different density estimation methods. The true exponential PDF (black), KDE (blue), Adaptive KDE (red), GMM approximation (green), QHBS (pink), One-point approximation (orange), and our method (cyan). The results for different sample sizes are shown from left to right: 300 samples (**a**), 100 samples (**b**), and 50 samples (**c**).

Table 10. Performance metrics for different methods for the Exponential distribution using 300 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	15.6920	5.3070	0.9526	0.0003	0.2042	0.0197
Adaptive KDE	15.0852	5.3081	0.9716	0.0009	0.1705	0.0158
GMM	9.3181	5.3093	0.9447	0.0003	0.2478	0.0209
QHBS	10.2243	5.3060	0.9769	0.0004	0.1755	0.0135
One-Point Padé	22.2235	5.3160	0.8965	**0.0002**	0.3207	0.0281
Proposed Method	**4.5099**	**5.3035**	**0.9819**	0.0013	**0.0891**	**0.0120**

Table 11. Performance metrics for different methods for the Exponential distribution using 100 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	12.0407	5.3112	0.9802	0.0004	0.2441	0.0128
Adaptive KDE	12.6357	5.3144	0.9461	0.0016	0.1867	0.0206
GMM	10.0232	5.3256	0.9134	0.0008	0.3727	0.0285
QHBS	9.8939	5.3125	0.9213	0.0003	0.2599	0.0295
One-Point Padé	**8.9574**	5.3130	0.9510	0.0015	0.2252	0.0226
Proposed Method	16.1897	**5.3065**	**0.9829**	**0.0003**	**0.1399**	**0.0119**

Table 12. Performance metrics for different methods for the Exponential distribution using 50 samples.

Method	**↓ WD**	**↓ BD**	**↑ CC**	**↓ KLD**	**↓ L1**	**↓ L2**
KDE	21.9270	5.3114	0.9694	0.0017	0.1903	0.0157
Adaptive KDE	14.0566	5.3204	0.9579	0.0106	0.2266	0.0195
GMM	20.7139	5.3246	0.9112	0.0009	0.3420	0.0334
QHBS	20.3703	5.3223	0.9123	0.0006	0.3835	0.0339
One-Point Padé	**13.4794**	5.3203	0.9296	0.0017	0.2398	0.0306
Proposed Method	16.5618	**5.3111**	**0.9897**	**0.0003**	**0.1702**	**0.0102**

6. Discussion and Conclusions

In this paper, we have considered the problem of estimating the PDF of observed data. Our novel scheme uses two powerful mathematical tools: the concept of moments and the relatively little-known Padé approximation. While moments incorporate crucial information which is central to both the time and frequency domains, the Padé approximation provides an effective means of obtaining convergent series from the data. Both of these phenomena are used in our new method to estimate the PDF in an inter-twined manner.

The method we propose is *nonparametric*. It invokes the concept of *matching* the moments of the original function and its learned PDF, and this, in turn is achieved by using the Padé approximation. Apart from the theoretical analysis, we have also experimentally evaluated the validity and efficiency of our scheme.

Although the Padé approximation is asymmetric, we have taken advantage of this asymmetry to our advantage. We have done this by working with two "mirrored" versions of the data so as to obtains different versions of the PDF. We have then effectively "superimposed" (or coalesced) them together to yield the final composite PDF. We are not aware of any other research that utilizes such a composite strategy, in any signal processing domain.

To evaluate the performance of the proposed method, we have employed synthetic samples obtained from various well-known distributions, including mixture densities. The accuracy of the proposed method has also been compared with that obtained by other approaches representative of the families of time- and frequency-domain methods available in the literature.

Our method has shown promising results across the distributions tested. However, we recognize the potential for further exploration. While our chosen examples demonstrate the method's efficacy, future work could involve testing on a broader range of distributions, including heavy-tailed and multimodal distributions. This extended testing could provide additional insights into the method's performance under diverse scenarios and potentially uncover new areas for refinement or application.

Moreover, in future research, we intend to explore methods that could enhance the robustness of the proposed approach, particularly in handling outliers. The current method, like other nonparametric density estimation techniques, may be sensitive to extreme values, as sample moments, which can be influenced by outliers. Although this issue can be mitigated through outlier detection and censoring prior to density estimation, we anticipate a potential for improvement. One promising direction would be to investigate the integration of more robust estimators for sample moments into the density estimation method. This could potentially increase the method's resilience to outliers, without compromising its performance on "clean" data, thereby expanding its applicability to datasets possessing more challenging characteristics.

The results underscore the robustness and effectiveness of our method, particularly in scenarios when the sample sizes are considerably reduced. Thus, this research confirms

how the state-of-the-art of estimating nonparametric PDFs can be enhanced by the Padé approximation, offering notable advantages over existing methods in terms of accuracy. As far as we know, the theoretical results that we have proven, and the experimental results that confirm them, are novel and rather pioneering.

Author Contributions: Conceptualization, S.A.H. and B.J.O.; methodology, all authors; software, H.R.A.; validation, all authors; writing—original draft preparation, H.R.A.; writing—review and editing, S.A.H., J.R.G., and B.J.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data used to generate the results in this study can be provided upon request to the authors.

Conflicts of Interest: The authors declare no conflicts 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. Pinsky, M.; Karlin, S. *An Introduction to Stochastic Modeling*; Academic Press: Cambridge, MA, USA, 2010.
2. Zilz, D.P.; Bell, M.R. *Statistical Results on the Performance of an Adaptive-Threshold Radar Detector in the Presence of Wireless Communications Interference Revision 1*; Technical Report 482; Purdue University School of Electrical and Computer Engineering: West Lafayette, IN, USA, 2017.
3. Tateo, A.; Miglietta, M.; Fedele, F.; Menegotto, M.; Pollice, A.; Bellotti, R. A statistical method based on the ensemble probability density function for the prediction of "Wind Days". *Atmos. Res.* **2019**, *216*, 106–116. [CrossRef]
4. Beitollahi, M.; Hosseini, S.A. Using curve fitting for spectral reflectance curves intervals in order to hyperspectral data compression. In Proceedings of the 2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Prague, Czech Republic, 20–22 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–5.
5. Beitollahi, M.; Hosseini, S.A. Using Savitsky-Golay filter and interval curve fitting in order to hyperspectral data compression. In Proceedings of the 2017 Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 2–4 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1967–1972.
6. Silverman, B.W. *Density Estimation for Statistics and Data Analysis*; Routledge: London, UK, 2018.
7. Chen, W.C.; Tareen, A.; Kinney, J.B. Density estimation on small datasets. *Phys. Rev. Lett.* **2018**, *121*, 160605. [CrossRef] [PubMed]
8. Kendall, M.G. *The Advanced Theory of Statistics*, 2nd ed.; Charles Griffin and Co., Ltd.: London, UK, 1946.
9. Hwang, J.N.; Lay, S.R.; Lippman, A. Nonparametric multivariate density estimation: A comparative study. *IEEE Trans. Signal Process.* **1994**, *42*, 2795–2810. [CrossRef]
10. Archambeau, C.; Verleysen, M. Fully nonparametric probability density function estimation with finite gaussian mixture models. In Proceedings of the 5th International Conference on Advances in Pattern Recognition, Calcultta, India, 10–13 December 2003; pp. 81–84.
11. Clerc, S.; Dennison, J.R.; Hoffmann, R.; Abbott, J. On the computation of secondary electron emission models. *IEEE Trans. Plasma Sci.* **2006**, *34*, 2219–2225. [CrossRef]
12. Gentle, J.E. Estimation of Probability Density Functions Using Parametric Models. In *Computational Statistics*; Springer: New York, NY, USA, 2009; pp. 475–485.
13. April, S. *Density Estimation for Statistics and Data Analysis Chapter 1 and 2*; Chapman and Hall: London, UK, 2003.
14. Hsu, C.Y.; Shao, L.J.; Tseng, K.K.; Huang, W.T. Moon image segmentation with a new mixture histogram model. *Enterp. Inf. Syst.* **2021**, *15*, 1046–1069. [CrossRef]
15. Reyes, M.; Francisco-Fernández, M.; Cao, R. Bandwidth selection in kernel density estimation for interval-grouped data. *Test* **2017**, *26*, 527–545. [CrossRef]
16. Knuth, K.H. Optimal data-based binning for histograms. *arXiv* **2006**, arXiv:physics/0605197.
17. Papkov, G.I.; Scott, D.W. aaaLocal-moment nonparametric density estimation of pre-binned data. *Comput. Stat. Data Anal.* **2010**, *54*, 3421–3429. [CrossRef]
18. Tsuruta, Y.; Sagae, M. Properties for circular nonparametric regressions by von Miese and wrapped Cauchy kernels. *Bull. Inform. Cybern.* **2018**, *50*, 1–13. [CrossRef]
19. Zambom, A.Z.; Ronaldo, D. A review of kernel density estimation with applications to econometrics. *Int. Econom. Rev.* **2013**, *5*, 20–42.

20. Demir, S. Adaptive kernel density estimation with generalized least square cross-validation. *Hacet. J. Math. Stat.* **2019**, *48*, 616–625. [CrossRef]
21. Hamsa, K. Some Contributions to Renewal Density Estimation. Ph.D. Thesis, University of Calicut, Kozhikode, India, 2009.
22. Jarosz, W. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*; University of California: San Diego, CA, USA, 2008.
23. Lagha, K.; Adjabi, S. Non parametric sequential estimation of the probability density function by orthogonal series. *Commun. Stat.-Theory Methods* **2017**, *46*, 5941–5955. [CrossRef]
24. Amindavar, H.; Ritcey, J.A. Padé approximations of probability density functions. *IEEE Trans. Aerosp. Electron. Syst.* **1994**, *30*, 416–424. [CrossRef]
25. Davies, K.; Dembińska, A. Computing moments of discrete order statistics from non-identical distributions. *J. Comput. Appl. Math.* **2018**, *328*, 340–354. [CrossRef]
26. Gamsha, A.M.; Hameed, M.A.; Roslan, R. Psolutions of poiseuille flow using homotopy perturbation method linked with Padé approximation. *ARPN J. Eng. Appl. Sci.* **2017**, *12*, 2545–2551.
27. Hosseini, S.A.; Ghassemian, H. Rational function approximation for feature reduction in hyperspectral data. *Remote Sens. Lett.* **2016**, *7*, 101–110. [CrossRef]
28. Singh, V.P.; Chandra, D. Reduction of discrete interval systems based on pole clustering and improved Padé approximation: A computer-aided approach. *Adv. Model. Optim.* **2012**, *14*, 45–56.
29. Gilewicz, J. Numerical detection of the best Padé approximant and determination of the Fourier coefficients of insufficiently sampled functions. In *Padé Approximants and Their Applications*; Springer: Berlin/Heidelberg, Germany, 1973; pp. 99–103.
30. Olver, S.; Townsend, A. Fast inverse transform sampling in one and two dimensions. *arXiv* **2013**, arXiv:1307.1223.
31. Jeffrey, M.R. Smoothing tautologies, hidden dynamics, and sigmoid asymptotics for piecewise smooth systems. *Chaos Interdiscip. J. Nonlinear Sci.* **2015**, *25*, 103125. [CrossRef] [PubMed]
32. Scott, D.W. Scott's rule. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 497–502. [CrossRef]
33. Upcroft, B.; Kumar, S.; Ridley, M.; Ong, L.L.; Durrant-Whyte, H. Fast re-parameterisation of Gaussian mixture models for robotics applications. In Proceedings of the Australasian Conference on Robotics and Automation, Canberra, Australia, 6–8 December 2004.
34. Kasa, S.R.; Rajan, V. Avoiding inferior clusterings with misspecified Gaussian mixture models. *Sci. Rep.* **2023**, *13*, 19164. [CrossRef]
35. Mas'ud, A.A.; Sundaram, A.; Ardila-Rey, J.A.; Schurch, R.; Muhammad-Sukki, F.; Bani, N.A. Application of the Gaussian Mixture Model to classify stages of electrical tree growth in epoxy resin. *Sensors* **2021**, *21*, 2562. [CrossRef] [PubMed]
36. Hosseini, S.A.; Ghassemian, H. Hyperspectral data feature extraction using rational function curve fitting. *Int. J. Pattern Recognit. Artif. Intell.* **2016**, *30*, 1650001. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Data Augmentation for Voiceprint Recognition Using Generative Adversarial Networks

Yao-San Lin [1,*], Hung-Yu Chen [2], Mei-Ling Huang [1,*] and Tsung-Yu Hsieh [1]

[1] Department of Industrial Engineering and Management, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 411, Taiwan; s4b215027@student.ncut.edu.tw
[2] Department of Information Management, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 411, Taiwan; chenhy@ncut.edu.tw
* Correspondence: yslin@ncut.edu.tw (Y.-S.L.); huangml@ncut.edu.tw (M.-L.H.)

Abstract: Voiceprint recognition systems often face challenges related to limited and diverse datasets, which hinder their performance and generalization capabilities. This study proposes a novel approach that integrates generative adversarial networks (GANs) for data augmentation and convolutional neural networks (CNNs) with mel-frequency cepstral coefficients (MFCCs) for voiceprint classification. Experimental results demonstrate that the proposed methodology improves recognition accuracy by up to 15% in low-resource scenarios. The optimal ratio of real-to-GAN-generated samples was determined to be 3:2, which balanced dataset diversity and model performance. In specific cases, the model achieved an accuracy of 96.6%, showcasing its effectiveness in capturing unique voice characteristics while mitigating overfitting. These results highlight the potential of combining GAN-augmented data and CNN-based classification to enhance voiceprint recognition in diverse and resource-constrained environments.

Keywords: data augmentation; deep learning; generative adversarial networks; GANs; information security; voiceprint recognition

1. Introduction

In this era of rapid technological advancement, artificial intelligence has become an essential part of daily life, offering various conveniences. A notable application of Artificial Intelligence (AI) is voice recognition, which has become ubiquitous in many real-world scenarios. Voiceprint recognition, in particular, is a key technology used for identity verification and has been extensively integrated into access control systems and activation devices.

Voiceprint recognition systems extract distinct features from speech signals to identify speakers. This technology is widely employed in security authentication and customer service. However, traditional voiceprint recognition systems often face challenges associated with small-sample learning. In many practical applications, the availability of sound samples is inconsistent or insufficient due to various constraints, which can result in suboptimal model performance. Furthermore, a lack of diverse training samples would limit the model's ability to recognize different patterns.

In real-world situations, collecting sound samples is difficult due to varying conditions, leading to an uneven and inadequate sample distribution. Furthermore, accurate sound sample collection is critical for effectively operating applications like access control systems.

To address these challenges, this study aims to improve the reliability, precision, and flexibility of voiceprint recognition systems, even when dealing with limited sample sizes. By employing generative adversarial networks (GANs)—a type of neural network capable of generating data similar to the original dataset—the study seeks to create synthetic sound samples. This approach aims to expand the dataset, particularly when certain samples are

scarce. By supplementing the training data with high-quality, diverse samples, the study anticipates improving the accuracy and dependability of the model.

Users can customize the GAN model to generate new samples tailored to specific requirements, allowing the model to train alongside the original dataset. This process enhances the system's ability to identify sounds in various real-world conditions, thereby reducing the cost and effort of collecting sound samples for companies developing similar systems. Utilizing GANs to generate new samples creates data that closely resemble real-world sound conditions. GANs effectively address challenges posed by diverse accents and languages, enabling voiceprint recognition systems to perform reliably across a wide range of linguistic and contextual variations. This approach not only reduces development costs but also eliminates the need for significant investments in collecting a wide variety of sound samples, making voiceprint recognition systems more accessible and reliable.

This study proposes an innovative approach to voiceprint recognition by integrating GANs and convolutional neural networks (CNNs) with mel-frequency cepstral coefficients (MFCCs) for enhanced data augmentation and feature extraction. We introduced a dynamic data augmentation strategy using GANs to generate synthetic voice samples, addressing the challenge of data scarcity. Second, the methodology incorporates an innovative iterative reduction technique to balance the ratio of real and GAN-generated samples, optimizing model performance and mitigating overfitting. Third, a custom-designed convolutional neural network (CNN) leverages MFCC features to effectively capture unique voice characteristics, ensuring high accuracy in resource-constrained scenarios. Finally, the approach is validated through comprehensive experiments on diverse datasets, demonstrating significant improvements in recognition accuracy and robustness compared to traditional methods.

By leveraging the power of GANs, we aim to create diverse and realistic voice samples that can enhance the performance of voiceprint recognition models, even in the face of limited training data. Generative adversarial networks have emerged as a powerful class of deep learning models that can generate novel and realistic synthetic data. These models consist of two neural networks, a generator, and a discriminator, that are trained in an adversarial fashion. The methodology enhances both data augmentation and feature extraction processes. The use of GANs allows for the generation of synthetic voice samples, effectively expanding the dataset and improving model training. A key innovation of this approach lies in the iterative reduction of synthetic samples, which balances the contribution of real and GAN-generated data to mitigate overfitting and maintain robust performance.

The CNN model is specifically tailored for voiceprint recognition tasks, employing MFCC features to capture critical speech characteristics. Designed with advanced techniques such as dropout layers and adaptive optimizers, the architecture is optimized to achieve high accuracy in resource-constrained environments. The methodology's effectiveness is demonstrated through comprehensive experiments involving diverse datasets, varied sample sizes, and different ratios of synthetic and real samples, showcasing its adaptability and reliability.

This approach also highlights its practical applications, including multilingual speaker verification, biometric authentication, and forensic analysis. By reducing data collection costs while maintaining high performance, the methodology offers a resource-efficient solution that is well-suited for real-world implementations. These contributions distinguish the proposed method from existing approaches, addressing the common limitations of data scarcity and overfitting in voiceprint recognition systems.

2. Literature Review

In the field of sound processing, insufficient samples are a major problem affecting model performance. The collection and annotation of sound data usually requires a lot of manpower and material resources, and GANs provide an effective method to generate more sound samples, so as to improve the combination of CNNs and MFCCs in sound-processing tasks. In this article, we will discuss the research on the use of GANs to generate samples

to solve the problem of insufficient sound samples processed by CNNs, and introduce the functions and operating principles of various tools.

The following section delves into the background and existing research related to the development of voice recognition systems, including CNNs, MFCCs, and the application of GANs in sound processing. These technologies are instrumental in overcoming challenges in voice recognition, particularly those posed by limited and diverse datasets.

2.1. Convolutional Neural Network for Voiceprint Recognition

Convolutional neural networks are a class of deep learning models primarily designed for tasks involving structured grid-like data, such as images and videos. CNNs have shown remarkable performance in tasks like image classification, object detection, and even speech recognition, when combined with certain preprocessing techniques like MFCCs. The distinguishing feature of CNNs is their ability to automatically and adaptively learn spatial hierarchies of features from input data. They consist of multiple layers, including convolutional layers, activation layers, pooling layers, and fully connected layers, each of which plays a specific role in feature extraction and classification.

A CNN is a powerful deep learning framework that excels in identifying structured patterns within data. While originally designed for image processing, CNNs have proven highly effective in speech and voice recognition tasks by leveraging feature representations such as spectrograms and MFCCs.

In this study, CNNs are utilized to process MFCC features extracted from voice samples. The CNN architecture comprises convolutional layers that detect intricate patterns within the MFCCs, pooling layers that reduce spatial dimensions while retaining critical features, and fully connected layers that perform final classification. The inclusion of dropout layers mitigates overfitting, ensuring the model's robustness. This design allows the network to capture unique voice characteristics, enabling precise speaker identification and verification.

By focusing on extracting and classifying voice features, the CNNs in this methodology are optimized for performance in low-resource and diverse voiceprint recognition scenarios. The model's parameters, such as the number of convolutional filters and kernel sizes, were selected based on empirical testing to balance computational efficiency and recognition accuracy.

2.2. Mel-Frequency Cepstral Coefficients in Voiceprint Recognition

Mel-frequency cepstral coefficients are widely used in speech and voice recognition for their ability to capture the essential characteristics of speech signals in a compact and computationally efficient form. By simulating the human auditory perception of sound frequencies, MFCCs provide robust features that enhance voiceprint recognition accuracy.

In this study, MFCCs are extracted from audio data to serve as input features for the CNN model. The preprocessing involves transforming raw audio signals into MFCC features using techniques such as frame segmentation, spectral analysis, and cepstral transformation. This ensures that critical information such as phonetic patterns and vocal characteristics is preserved for subsequent classification.

Delta and delta-delta cepstrum features, which capture the rate of change and temporal dynamics of the MFCCs, are also included to enhance the model's ability to generalize across diverse speech patterns. By combining static and dynamic features, the MFCC representation allows the CNN to learn both the fundamental and evolving attributes of voice samples, improving speaker recognition performance even with limited data.

2.3. Generative Adversarial Networks

GANs are a type of neural network architecture designed for unsupervised learning, consisting of two main components: a generator and a discriminator. These networks are trained in a competitive setting, where the generator tries to create data similar to the real dataset, and the discriminator evaluates whether the data are real or generated.

Through this adversarial process, GANs can generate realistic data that mimic the statistical distribution of the training dataset, making them highly effective in tasks like image generation, speech synthesis, and data augmentation.

2.3.1. Generator

The generator in a GAN is responsible for creating synthetic data that resemble real data. It starts with random noise as input, typically sampled from a uniform or Gaussian distribution, and transforms this noise into structured data through a series of layers. The generator's goal is to produce outputs that are indistinguishable from real data.

The process begins by sampling from a latent space, which is a lower-dimensional representation of the input space. The latent space captures the underlying factors that define the data distribution. By feeding noise into the latent space, the generator produces output through a series of transformations, such as fully connected layers, convolutional layers, and transposed convolutions (for images). These transformations allow the generator to map random noise to structured data, like images, audio, or text.

The generator's objective is to "trick" the discriminator into classifying its output as real. This goal is achieved by iteratively updating the weights of the generator through backpropagation, based on feedback from the discriminator. The better the generator becomes at fooling the discriminator, the closer the generated data becomes to the real data distribution.

Formally, the generator's objective is to minimize the following loss function, as follows:

$$L_G = -\mathrm{E}_{z \sim p_z(z)}[\log(D(G(z)))]$$

where $G(z)$ is the output of the generator given input noise z, and $D(G(z))$ is the discriminator's prediction for the generated data.

The goal is to minimize the discriminator's ability to distinguish between real and fake data.

2.3.2. Discriminator

The discriminator is a binary classifier that distinguishes between real data (from the training set) and fake data (produced by the generator). Its job is to correctly label real data as real and generated data as fake. The discriminator receives inputs from both the real dataset and the generator's output and must decide whether each input is genuine or artificially created.

The discriminator is typically implemented as a neural network, which can consist of fully connected layers or convolutional layers, depending on the task (e.g., image, text, or audio recognition). It assigns a probability score to each input, indicating whether it believes the input is real (closer to 1) or generated (closer to 0). The discriminator's training process involves minimizing its classification error by correctly identifying real versus fake data.

The discriminator's objective function is to maximize the likelihood of correctly classifying real and fake samples. This can be expressed as follows:

$$L_D = -\mathrm{E}_{x \sim p_{data}(x)}[\log(D(x))] - \mathrm{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

where $D(x)$ is the discriminator's prediction for a real sample x, and $D(G(z))$ is its prediction for a generated sample $G(z)$.

The discriminator aims to maximize its ability to distinguish between real and fake data while minimizing the generator's chances of fooling it. As the discriminator improves, it becomes more challenging for the generator to produce convincing fake data.

2.3.3. Objective Function

GANs use an adversarial training process, where the generator and discriminator are both trying to optimize opposing objectives. The discriminator seeks to maximize

the probability of correctly classifying real versus fake data, while the generator tries to minimize the discriminator's ability to tell the difference. This competition is modeled using a minimax objective function, as follows:

$$\min_G \max_D V(D,G) = \mathrm{E}_{x \sim p_{data}(x)}[\log(D(x))] + \mathrm{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

This equation represents the minimax game between the generator and the discriminator, as set out below:

- The generator (G) aims to minimize this function, by improving its ability to produce data that the discriminator cannot distinguish from real data.
- The discriminator (D) aims to maximize this function, by improving its ability to correctly identify real and generated data.

The objective function is based on the cross-entropy loss, commonly used in classification tasks. As the GAN trains, the generator's ability to produce realistic data improves, while the discriminator becomes more adept at distinguishing real data from fake data. The ideal state, or Nash equilibrium, is reached when the generator can produce data so realistic that the discriminator cannot tell the difference with better than random guessing (i.e., 50% accuracy).

GANs consist of two neural networks—a generator and a discriminator—that are trained simultaneously in an adversarial setup. The generator creates synthetic samples from random noise, while the discriminator evaluates whether the samples are real or generated. This iterative process improves the generator's ability to produce realistic data while enhancing the discriminator's ability to distinguish between real and synthetic samples.

The generator aims to minimize the discriminator's ability to differentiate between real and generated samples, while the discriminator strives to maximize its classification accuracy. This dynamic adversarial training is modeled as a minimax optimization problem, where the generator and discriminator iteratively improve by competing against each other. Effective training requires carefully tuned hyperparameters, such as learning rates and batch sizes, to ensure convergence and avoid instability or pattern collapse.

In this study, the GAN is utilized to augment the voiceprint dataset by generating synthetic audio samples. The training process incorporates mechanisms such as regularization and adaptive learning rates to stabilize the adversarial training and maintain the diversity of generated samples. By combining the synthetic samples with real data, the GAN enhances the training dataset, addressing the challenges of data scarcity and improving the performance of the voiceprint recognition system.

2.4. Application of Deep Learning in Speaker Recognition

GANs have seen extensive application in various fields, especially in tasks involving data generation and enhancement. Research in deep learning and speech processing has demonstrated that GANs can address significant challenges in areas like speaker recognition, speech enhancement, and data augmentation. In this section, we explore the role of GANs in conjunction with other deep learning techniques and spectral feature extraction methods, as well as their application in generating and enhancing speech data.

Wan et al. proposed a method for extracting speaker features using deep neural networks, called Deep Speaker Embeddings. These embeddings effectively distinguish between different speakers, significantly enhancing the accuracy and stability of speech recognition [1]. Kang et al. introduced a CNN-based end-to-end speaker verification method for text-independent vocal verification by learning speech feature representations. This approach demonstrates good adaptability and robustness when processing different speech datasets, while also showcasing the application of CNNs in speech feature extraction, which could serve as a reference for combining MFCC and a GAN to generate samples [2]. Snyder et al. presented a method for speaker recognition using deep neural network (DNN) embeddings. This method extracts features through a DNN for vocal recognition, leading to a significant performance boost [3].

Chung and Nagrani introduced the VoxCeleb2 dataset to demonstrate its use in deep voice recognition, showing a notable improvement in voice recognition performance. VoxCeleb2 contains a large variety of speech data, which contribute to advancing deep learning research in this field [4]. Lei et al. proposed a voice recognition model based on a phoneme perception deep network. By incorporating phonemic information, this model improves its ability to differentiate vocal features, thus enhancing recognition accuracy [5]. Ravanelli and Bengio introduced SincNet, a deep network model for speaker recognition directly from raw waveforms. SincNet shows strong performance by extracting features directly from the original audio waveform, avoiding complex pre-processing steps [6]. Heigold et al. developed an end-to-end text-dependent speaker verification method that extracts features from speech data using a DNN. This approach simplifies traditional feature extraction and matching steps, improving verification accuracy through end-to-end learning [7].

Snyder et al. also proposed another text-independent speaker verification method using DNN embeddings, significantly enhancing verification performance. This approach emphasizes the robustness and consistency of DNN embeddings in handling different textual content [3].

2.5. Spectral Feature Extraction Methods

Spectral feature extraction is a widely used technique in speaker recognition, with MFCCs being one of the most common features. Hinton et al. summarized common views on the role of DNNs in speech modeling, emphasizing the importance of spectral features in deep learning [8]. Davis and Mermelstein conducted a detailed comparison of multiple parametric representations and proposed the use of MFCCs in continuous speech recognition. Their method converts speech signals into feature vectors, making the subsequent recognition process more efficient and accurate [9].

Rabiner and Juang extensively covered the basics of speech recognition, including spectral feature extraction methods, in their work. They delved into techniques like linear predictive cepstral coefficients (LPCCs) and MFCCs, comparing their performance in various speech recognition tasks [10].

Muda et al. introduced an algorithm combining MFCC and dynamic time warping (DTW) for speech recognition, effectively improving recognition efficiency by leveraging MFCC's feature extraction capabilities and DTW's time alignment [11].

Logan discussed the application of MFCCs in music modeling, showing that MFCCs are effective across a range of audio processing tasks, improving the accuracy and stability of feature extraction [12].

Kim and Stern introduced power normalized cepstral coefficients (PNCCs), a new spectral feature extraction method that offers better robustness in noisy environments [13]. Wang et al. proposed a line spectrum frequency distance measurement method based on one-way inner product for speech feature extraction. This new method improves the accuracy and stability of feature extraction while effectively handling background noise [14]. Alkhatib et al. researched the principles of MFCC and LPCC parameter extraction, and the methods for extracting differential cepstral parameters. Their study demonstrated that using MFCC, LPCC, and their first- and second-order differences in combination with the K-means algorithm and a three-layer backpropagation neural network can significantly improve recognition rates. They also confirmed that MFCC parameters offer better robustness than LPCC [15].

2.6. Application of Generative Adversarial Networks in Speech Synthesis and Enhancement

GANs have shown great potential in speech synthesis and enhancement. Saito et al. explored GAN-based methods for speech enhancement and noise robustness in vocal verification [16]. Wang et al. discussed the use of GANs for data augmentation in speaker recognition. Their approach significantly improves recognition performance by generating new training samples, particularly when data are insufficient [17]. Pascual et al. proposed

the speech enhancement generative adversarial network (SEGAN), which reduces noise and enhances speech quality through adversarial training, generating clearer and more natural speech signals [18].

Sisman et al. summarized the development and challenges in speech conversion research, exploring various GAN applications and outlining the technical challenges faced in this field [19]. In Kaneko et al.'s study, an innovative speech conversion method was proposed using a sequence-to-sequence (Seq2Seq) model with GAN to learn similarity measures, improving the naturalness of generated speech [20]. Chandna et al. discussed CNN-based techniques for mono audio source separation, which was particularly useful in speech enhancement tasks as it effectively separated speech from background noise [21].

Skerry-Ryan et al. introduced a GAN-based Tacotron model for intonation transfer, achieving more natural speech synthesis effects by adapting intonation characteristics to different contexts and emotional expressions [22]. Pan et al. proposed a GAN with dual discriminators for speech enhancement, improving the quality and stability of generated speech, particularly in noisy environments [23].

In addition to deep learning and spectral feature extraction, traditional machine learning methods like support vector machines (SVMs) and Gaussian mixture models (GMMs) are still widely applied in speaker recognition. Campbell et al. combined SVM and GMM hypervectors for speaker verification, demonstrating improved performance in discriminative recognition [24].

Reynolds et al. detailed the use of adaptive GMMs in speaker verification, showcasing the model's effectiveness and stability in this field [25]. Bimbot et al. provided a thorough review of text-independent speaker recognition techniques, serving as a valuable resource for researchers [26]. Kinnunen and Li summarized recent research advances in speaker recognition, covering topics from feature extraction to hypervector modeling and identifying key challenges [27].

Mak and Yu analyzed various voice activity detection techniques for the National Institute of Standards and Technology (NIST) used in NIST's speaker recognition evaluations, providing experimental data on their performance in noisy environments [28]. Burget et al. examined the role of feature extraction and channel compensation in GMM-based speaker recognition systems, proposing several methods for improving performance [29].

While the reviewed studies demonstrate significant advancements in speaker recognition and speech processing, there are notable limitations. Many of the approaches are heavily reliant on deep learning models, which can require large amounts of computational resources and data for training. For instance, the work by Wan et al. on Deep Speaker Embeddings [1] and Kang et al. on CNN-based speaker verification [2] showcase the effectiveness of DNNs but highlight the issue of data scarcity, which GANs partially address but do not completely resolve. Additionally, Snyder et al.'s [3] DNN embedding method significantly boosts performance but raises concerns about the model's robustness in diverse real-world conditions. Chung and Nagrani's work on the VoxCeleb2 dataset [4] emphasizes the importance of large datasets, but such resources are not always available, limiting the scalability of these solutions.

Moreover, traditional methods like SVM and GMM, though augmented by GAN-generated data [16,24,25], may lag behind deep learning models in performance, particularly in more complex environments. Lastly, although GANs have proven effective in data augmentation and noise reduction, as seen in SEGAN [18], their adversarial training process is sensitive to hyperparameters and can lead to instability in training, as noted by Pan et al. [23]. Therefore, while GANs and deep learning have driven significant progress, challenges such as model scalability, data dependency, and training stability remain.

3. GAN-Enhanced CNN for Voiceprint Recognition

The methodology described here proposes an algorithm combining both CNN and GAN in an innovative way to address the challenge of limited data in voiceprint recognition, named GAN-enhanced CNN for voiceprint recognition (GEC-VR). The CNN is used

for classification, while the GAN model augments the dataset by generating new audio samples, thus enhancing the training process, as Figure 1 shows. The detailed description is as follows:

Input:

- Audio samples (e.g., from Common Voice dataset);
- Number of original samples N;
- Number of GAN-generated samples G;
- CNN model initialized.

Output:

- Trained CNN model;
- Accuracy metrics for voiceprint recognition;

Algorithm Steps:

1. Data Collection
   - Collect N audio samples, ensuring diversity in gender, language, and voice characteristics.
   - Split the data into training and test sets, where $N = N_{train} + N_{test}$:

     $N_{train} = 50$;
     $N_{test} = 10$.
2. Preprocessing
   - Convert all audio files into Waveform Audio File (WAV) format.
   - Extract MFCC from audio:

     Use librosa.feature.mfcc to compute MFCC features.
     Set $n_{\text{mfcc}} = 13$, i.e., extract 13 feature variables.
   - Format the MFCC features to ensure consistent input length for CNN.
3. CNN Model Construction
   - Initialize a CNN model with the following layers:

     Convolutional layers: Use four convolutional layers with ReLU activation.
     Pooling layers: Apply max-pooling after each convolutional layer.
     Fully connected layers: Include three fully connected layers for classification.
     Dropout layers: Introduce dropout between fully connected layers to prevent overfitting.
   - Compile the model using:

     Loss function: sparse_categorical_crossentropy;
     Optimizer: Adam.
4. GAN Model Construction
   - Generator:

     Generate new samples based on random noise (e.g., 600-dimensional noise).
   - Discriminator:

     Classify the generated samples as real or fake.
   - Use a GAN training cycle to improve the quality of the generated audio samples.
5. Training and Evaluation
   - Combine the original samples N with the GAN-generated samples GGG.
   - Train the CNN model:

     Perform forward propagation and backpropagation.
     Train for a predefined number of epochs (e.g., 30 epochs).
   - Evaluate model accuracy using the test set.
   - Repeat the training cycle multiple times (e.g., 50 rounds) to ensure stable performance.

6. Iterative GAN Reduction and Re-training
    - Reduce the number of GAN-generated samples by Δ. (Δ can be determined by user, e.g., five).
    - Re-train the model and evaluate accuracy.
    - Repeat the reduction and re-training process for 10 cycles.
    - Track accuracy across different GAN sample sizes to determine the optimal balance between real and GAN-generated data.
7. Final Evaluation
    - Record the final model's accuracy metrics for voiceprint recognition.
    - Analyze the impact of GAN sample generation on model performance.

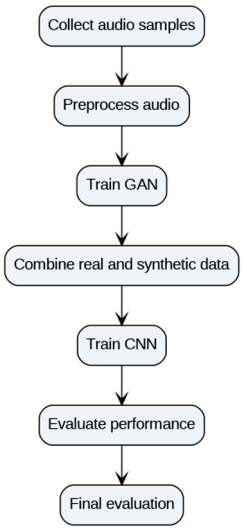


Figure 1. Algorithm flowchart.

3.1. Data Collection and Audio Preprocessing

The diversity of voiceprint data is essential for training a model with strong general capabilities. This includes characteristics such as language, dialect, gender, age, and intonation. A diverse dataset helps the model maintain a high level of recognizability across various real-world application scenarios. By learning from different datasets, the model can extract detailed features and improve identification accuracy. In this study, data were collected from the Common Voice database [30], which provides a wide variety of voices in different languages and genders for research use. Users can also contribute by uploading their own voice samples. During the modeling process, 10 cases were collected, with each case containing 60 sound clips ranging from 3 to 4 s.

The dataset used in this study consists of voice samples collected from 10 unique cases, with 60 original samples per case. To address the challenge of limited data diversity, GANs were employed to augment the dataset. For each case, GANs generated an additional 40 synthetic samples, resulting in a total of 100 samples per case after augmentation. The final dataset size across all cases is therefore 1000 samples, comprising 600 original and 400 GAN-generated samples. This augmentation ensured a balanced representation of diverse voice characteristics, enhancing the robustness of the model training process.

The pre-processing phase involves manually tagging the voice data into distinct categories, as the downloaded audio is untagged. Once tagged, the data are stored in separate folders for later use. Before building the model, the collected audio must be categorized and stored in specific folders where the code will be executed. The next step involves converting all audio files into WAV format to ensure consistent data input for the model. The modeling was carried out using Google Colab, which provided a cloud-based runtime environment. Zero-padding was applied to standardize the length of all audio

samples. This ensures uniform input dimensions for the CNN while preserving the original data integrity, avoiding truncation or distortion.

3.2. Feature Extraction—MFCC

Voiceprint recognition relies on extracting meaningful features from speech signals to identify individuals. In this process, feature extraction is crucial to the model's performance. These features include attributes like fundamental frequency, intensity, timbre, and tempo of speech. Such features help the model capture the unique voice characteristics, enhancing the accuracy of recognition. Effective feature extraction not only improves the model's ability to identify learned cases but also helps it generalize to new, unseen data, making the model more robust.

3.2.1. Audio Data Loading and the MFCC Characteristics Calculations

The 'librosa.load' function is used to load audio data, generating two primary outputs: an audio time series and the sample rate (the number of samples per second). The parameter 'sr = 22,050 Hz' was chosen for this study to determine the sample rate. This function decodes the audio file, making it ready for processing.

The 'librosa.feature.mfcc' function extracts the MFCC from the audio data. MFCC is a common feature used in speech recognition. For this study, the 'n_mfcc' parameter was set to 13, meaning that 13 MFCC feature variables were extracted from each audio sample.

3.2.2. Formatting

All extracted audio features were padded to the same length to prepare the data for CNN model training. Padding, rather than cropping, was used to ensure that no important features were lost. After padding, the data were reshaped into a four-dimensional array to meet the input requirements of the CNN. This process ensures that the features are complete and ready for training.

MFCCs were selected as the primary features for voiceprint recognition due to their established effectiveness in capturing speech characteristics. MFCCs simulate the human auditory system's perception of sound frequencies, making them particularly well-suited for distinguishing vocal patterns. They are widely used in speech and speaker recognition tasks because of their ability to represent both static and dynamic speech information when combined with delta and delta-delta coefficients.

While MFCCs offer robust feature extraction, alternative features like linear predictive cepstral coefficients (LPCCs) also have potential for voiceprint recognition. LPCCs emphasize speech production characteristics and could complement MFCCs to capture additional vocal nuances. However, this study focused exclusively on MFCCs to maintain a targeted scope and streamline the experimental design. Future work will explore the integration of LPCCs and other advanced acoustic features to further enhance model performance and generalizability.

3.3. CNN Model Construction

The voiceprint recognition model was built using pre-processed audio data in a graphics processing unit (GPU) environment on Google Colab, which significantly accelerated the training process compared to using tensor processing units (TPUs) or central processing units (CPUs). Pre-processed audio data refer to audio signals that have undergone format conversion, resampling, segmentation, MFCC extraction, and padding to ensure their consistency and suitability for CNN-based classification. The model's architecture relies on several key components to effectively process the audio features and make predictions.

The CNN used in this study is designed specifically for voiceprint recognition, leveraging MFCCs as input features. The model incorporates multiple convolutional and pooling layers to effectively extract and reduce the sampling rate of critical features from the MFCC representations. This architecture ensures the preservation of important voiceprint characteristics while reducing computational complexity.

The CNN consists of four convolutional layers, each followed by max-pooling layers to reduce feature dimensions and improve generalization. The convolutional layers use ReLU (rectified linear unit) as the activation function, which introduces non-linearity and ensures efficient gradient propagation during training. The number of filters in the convolutional layers progressively increases from 32 to 256, with a kernel size of 3×3 in all layers. These choices were determined through empirical testing to balance model complexity and accuracy.

The fully connected layers, placed after the convolutional and pooling stages, include three dense layers with 512, 256, and 7 neurons, respectively. The CNN architecture was originally designed with seven output nodes to accommodate a broader range of classes. For this study, the output layer was adjusted to five nodes to align with the dataset, which includes five speaker classes. This modification ensures consistency between the model and the dataset. Dropout layers with a rate of 0.5 are incorporated between the fully connected layers to prevent overfitting and ensure the model's robustness across diverse datasets. The output layer uses a softmax activation function to classify voiceprints into distinct categories.

The model is optimized using the Adam optimizer, chosen for its adaptive learning rate capabilities, which help stabilize training and enhance convergence. The sparse categorical crossentropy loss function is used, as it is well-suited for multi-class classification tasks. Training is conducted over 30 epochs with a batch size of 32, parameters chosen based on performance tuning to balance efficiency and model accuracy.

This architecture and hyperparameter design ensure that the CNN is effective for voiceprint recognition, particularly in scenarios with limited and diverse datasets. By combining efficient feature extraction and robust classification capabilities, the model achieves high accuracy in recognizing distinct voiceprints.

3.4. Model Training and Evaluation

Once the model construction was complete, it was trained to learn patterns from the data. Afterward, its adaptability and accuracy were tested on a separate test dataset. Accuracy, defined as the ratio of correct predictions to total predictions, was evaluated over 10 rounds of training, and the best-performing model was selected for further use. A standard CNN architecture was chosen to ensure the reliability of the classification results. This allowed us to isolate and evaluate the contributions of the iterative reduction technique without introducing confounding variables from complex or proprietary CNN designs.

The model's performance was optimized using the sparse_categorical_crossentropy loss function, which is particularly suited for multi-class classification tasks with integer labels. This loss function measures the difference between predicted and true labels, with the goal of minimizing this discrepancy to enhance the model's accuracy. To achieve this, the Adam optimizer was employed, which combines the strengths of AdaGrad and RMSProp. By calculating the first and second moment estimates of the gradient, Adam provides greater stability and requires less tuning compared to traditional optimization algorithms like the stochastic gradient descent (SGD). The CNN used in this study is a well-established architecture for classification tasks. It serves as a robust tool for processing the combined dataset of real and GAN-generated samples, supporting the implementation of the iterative reduction technique.

The process of training the model involves two main phases: forward propagation and backpropagation. In forward propagation, data flow through the network from the input layer to the output layer, with each neuron calculating output values based on the current weights. After this step, backpropagation is used to adjust the weights and biases by computing the gradient of the loss function with respect to each weight. The model then uses this information to update the weights, aiming to minimize the loss by applying the gradient descent algorithm or the Adam optimizer.

Training is conducted over multiple cycles, called epochs, where the entire dataset passes through the model. Each epoch includes a complete forward and backpropagation

cycle. In this study, 30 epochs were performed, meaning the model iterated over the data 30 times, refining its predictions and learning from the patterns in the data.

3.5. GAN Model Construction

GAN produces various samples through the trained generator and lets the discriminator determine whether the generated samples are real samples or fake samples; subsequently, the discriminator produces feedback. The audio quality generated by the adversarial generator gradually increases, and the discriminant ability of the discriminator gradually improves; this mechanism allows for samples generated by the GAN to have high-quality output. It can also help us improve the problem of low accuracy caused by insufficient samples when building voiceprint recognition models. The GAN model runs in the TPU environment of Colab, and the TPU is more efficient than the GPU and CPU for such a small but very large amount of data processing. To mitigate pattern collapse and instability, the Wasserstein GAN with gradient penalty (WGAN-GP) architecture was utilized. Additionally, hyperparameters such as learning rates and batch sizes were optimized, and diversity was monitored by visualizing and evaluating the statistical distribution of generated samples.

3.5.1. GAN Audio Preprocessing

Prior to training the GAN, audio samples are preprocessed to ensure consistency and compatibility with the model. The preprocessing steps are as follows:

1. Audio loading: the audio files are loaded using the torchaudio.load function, and the first channel is selected in case of multi-channel audio.
2. Standardized length: audio lengths are padded to match the longest sample in the dataset, ensuring uniformity.
3. Sample rate: a sample rate of 48,000 Hz is used to capture audio details effectively, as higher rates (e.g., 96,000 Hz) significantly increase computational costs with marginal gains in quality.

3.5.2. Generator Design

The GAN generator transforms random noise into synthetic audio samples through a series of layers, as follows:

1. Noise input: a 600-dimensional random noise vector serves as the input.
2. Layer expansion: the generator progressively increases the feature dimensions through fully connected layers, from 256 up to 4096 dimensions.
3. Activation functions:

- LeakyReLU is introduced after each intermediate layer to prevent vanishing gradients and ensure nonlinearity.
- Tanh is applied in the final layer to generate outputs in the range of $[-1, 1]$, simulating real audio signals.

The generator's design ensures high-quality audio generation while balancing computational efficiency.

3.5.3. Discriminator Design

The discriminator evaluates the authenticity of audio samples. Its structure mirrors the generator but operates in reverse, as follows:

1. Input dimension: takes a flattened input of 4096 dimensions (or reshaped as (64, 64, 1)).
2. Layer reduction: successively reduces the feature dimensions from 4096 to 256.
3. Activation functions:

- LeakyReLU maintains gradient flow for efficient learning.
- Sigmoid outputs a probability indicating whether a sample is real (close to 1) or synthetic (close to 0).

The discriminator and generator jointly refine their performance during adversarial training.

The generator's goal is to trick the discriminator into judging the generated audio samples as real. Therefore, the loss of the generator is the cross-entropy between the output of the discriminator on the generated sample and the target label 1 (true). For the transformation process, the generator in the GAN model performs the following transformations. First, the initial noise vector (latent_dim = 600) serves as a random input to the generator. The generator applies multiple fully connected and convolutional layers to increase the dimensionality of the data, producing a flattened vector of size 4096. This output vector is reshaped into a matrix of dimensions (64, 64, 1), simulating a single-channel feature map. The GAN produces a flattened vector of size 4096. This output vector is reshaped into a matrix of dimensions (64, 64, 1) to match the input requirements of the CNN model, as Figure 2 shows.

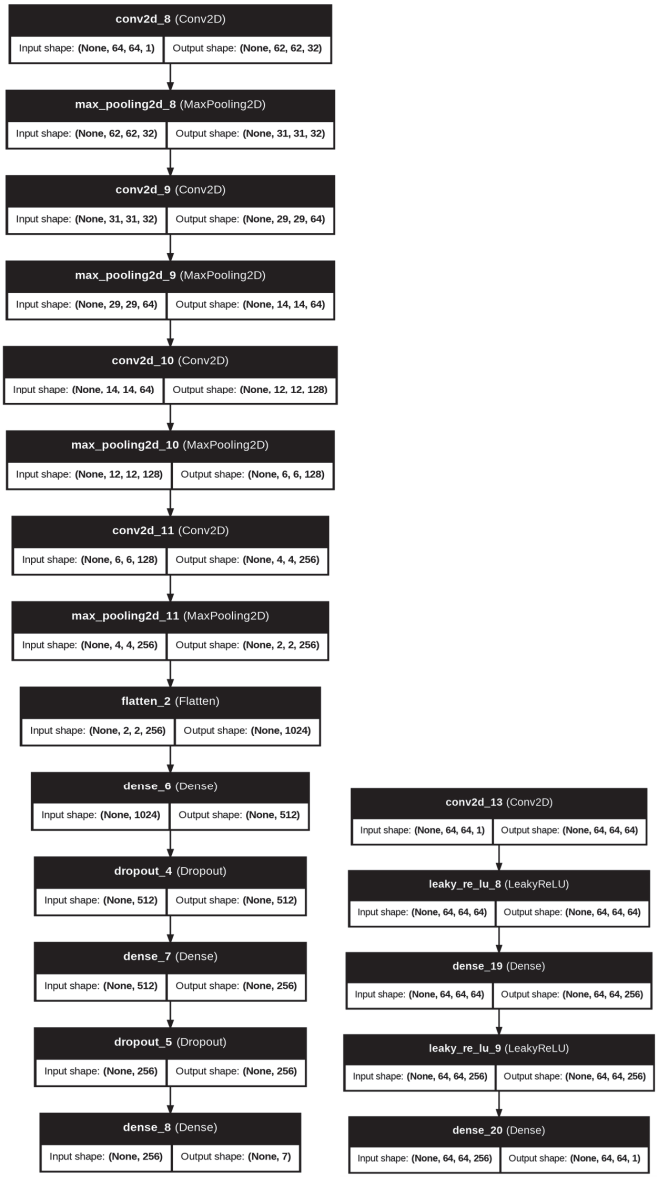


Figure 2. CNN architecture. The output layer was configured with 5 nodes to classify the 5 speaker classes used in this study. (**Left**) Illustration of the CNN architecture used in the study. The input dimension (64, 64, 1) represents the reshaped MFCC feature matrix derived from audio samples, while the output value of 7 corresponds to the number of speaker classes in the classification task. (**Right**) GAN model training process for voiceprint recognition.

This transformation process bridges the gap between the generator's latent space and the input requirements of the CNN, enabling the end-to-end integration of GAN-augmented data into the classification pipeline.

3.5.4. GAN Reliability Test

After the GAN model is built, the next step is to generate audio, and use the voiceprint recognition model built earlier to test the accuracy. After 10 constructions and trainings, the models with the highest accuracy were compared one-by-one for further use.

One key aspect of GEC-VR is the iterative reduction of GAN-generated samples to find the optimal number of synthetic data points that improve the model's performance without causing overfitting. By adjusting the number of GAN samples, the algorithm can be expected to ensure that the model generalizes well to unseen data.

Adaptive optimizers like Adam, combined with MFCC feature extraction, improve the efficiency and accuracy of training voiceprint recognition models by leveraging advanced signal processing and optimization techniques. However, careful monitoring of GAN over-reliance is necessary, as generating too many synthetic samples can degrade model performance. This algorithm provides a balanced and systematic approach that addresses real-world challenges in voiceprint recognition with limited and diverse datasets.

Traditional GAN evaluation metrics, such as Fréchet inception distance (FID), are designed for visual domains and lack standardization for evaluating synthetic audio. Consequently, this study assesses the GAN indirectly by analyzing its contribution to the voiceprint recognition system's performance, measured through accuracy, precision, recall, and F1 score.

4. Experimental Studies

4.1. Data Processing

To ensure consistency during the evaluation, all original voice data were divided into a training set and a test set. This step was crucial to guarantee that accuracy changes in different cases could be evaluated without the randomness of the test set affecting the results. Each case dataset comprised 60 samples, with 50 samples allocated for training and 10 for testing. The analysis began with five cases: four male voices and one female voice. Table 1 shows the training and testing sample distribution for each case while Table 2 shows the confusion matrix of the texting set.

Table 1. Number of training and testing samples of each case dataset. In this table, 'Cases' denote individual speaker classes in the dataset (e.g., Case 1 corresponds to Speaker A, Case 2 to Speaker B, etc.).

Datasets	Training Samples	Testing Samples
Case 1	50	10
Case 2	50	10
Case 3	50	10
Case 4	50	10
Case 5	50	10

Note: The CNN was originally designed with seven output nodes to accommodate potential future expansions. For this study, the architecture was adjusted to five output nodes to match the dataset, which included five speaker classes (four male and one female).

Table 2. Confusion matrix of testing set.

Actual Category/Forecast Results	Case 1	Case 2	Case 3	Case 4	Case 5
Case 1	8	0	0	0	2
Case 2	0	10	0	0	0
Case 3	0	0	10	0	0
Case 4	0	0	0	10	0
Case 5	0	0	2	0	8

Note: The CNN was originally designed with seven output nodes to accommodate potential future expansions. For this study, the architecture was adjusted to five output nodes to match the dataset, which included five speaker classes (four male and one female).

4.2. GAN Generation and Training

Taking Case 1 as an example, the 50 training samples were reduced to 10 samples. Using the remaining 10 samples, 60 new samples were generated by the GAN. After combining these 60 new samples with the old ones, the model was trained using the previously built CNN model, and the accuracy was tested after 50 rounds of repeated training. Table 3 presents the new sample count for Case 1 (10 original + 60 GAN-generated samples) compared with other cases, which retained 50 original samples. While 100% training accuracy is theoretically achievable, particularly in small or well-separated datasets, it warrants careful examination to rule out overfitting or bias. Metrics such as precision, recall, and F1 score are calculated per class and can show variations even when the overall accuracy is 100%. This occurs due to minor discrepancies in class-level performance. In this study, 100% accuracy was observed in specific cases due to several factors, as described below.

Table 3. Performance evaluation based on Table 2, incorporating precision, recall, and F1 score metrics. Precision, recall, and F1 score are calculated per class and can show variations even when the overall accuracy is 100%. This occurs due to minor discrepancies in class-level performance.

Datasets (Case)	Precision (%)	Recall (%)	F1-Score (%)
1	80.0	80.0	80.0
2	100	100	100
3	83.3	100	90.9
4	100	100	100
5	80.0	80.0	80

Note: The CNN was originally designed with seven output nodes to accommodate potential future expansions. For this study, the architecture was adjusted to five output nodes to match the dataset, which included five speaker classes (four male and one female).

Dataset characteristics: The dataset size per case was relatively small (100 samples, including 60 real and 40 GAN-generated), and the samples within each case exhibited distinct voiceprint patterns. This led to highly separable classes in the training data.

Regularization techniques: To mitigate overfitting, dropout layers were incorporated into the CNN model, ensuring robust generalization. Additionally, GAN-generated samples added diversity to the training set, improving the model's capacity to learn generalizable patterns.

Balanced training: by carefully tuning the ratio of real to synthetic samples, the model avoided overfitting to either data type, resulting in a stable training performance.

Confusion matrices were generated for each training step, showing the prediction results for each case. For instance, Table 2 shows that Case 1 had two misidentifications as Case 5, while Cases 2, 3, and 4 had no errors. Based on this, the accuracy for Case 1 was 80%, as outlined in Table 3, which summarizes the accuracy per training session. Table 4 shows the averaged accuracies of training on hybrid samples.

Table 4. Averaged accuracies of training on hybrid samples (real data and generated samples) for 50 independent experiments.

Datasets	Number of Training Samples	Number of Testing Samples	Averaged Accuracy of 50 Experiments
Case 1	10 + 60 GAN	10	75.6%
Case 2	50	10	100%
Case 3	50	10	95.2%
Case 4	50	10	98%
Case 5	50	10	91.2%

Note: The CNN was originally designed with seven output nodes to accommodate potential future expansions. For this study, the architecture was adjusted to five output nodes to match the dataset, which included five speaker classes (four male and one female).

4.3. Repetitive Training Analysis

To more clearly observe the effects of different GAN sample sizes on accuracy, the number of GAN-generated samples was progressively reduced from 60 to 5, with each reduction followed by 50 training sessions. The average accuracy was recorded after each reduction. Table 5 illustrates how the accuracy varied depending on the number of GAN samples generated for Case 1. The results indicate that accuracy initially improved with more GAN samples but then declined, possibly due to the model overfitting to the generated data, as Figure 3 shows.

Table 5. Number of generated samples and corresponding average accuracy for Case 1.

Case 1: Number of Samples	Averaged Accuracy of 50 Independent Experiments (%)
10 (initial template)	96.6
15 (5 GAN)	52.8
20 (10 GAN)	81.6
25 (15 GAN)	94.6
30 (20 GAN)	92.2
35 (25 GAN)	93
40 (30 GAN)	96.6
45 (35 GAN)	92.2
50 (40 GAN)	69.4
55 (45 GAN)	86
60 (50 GAN)	86.4

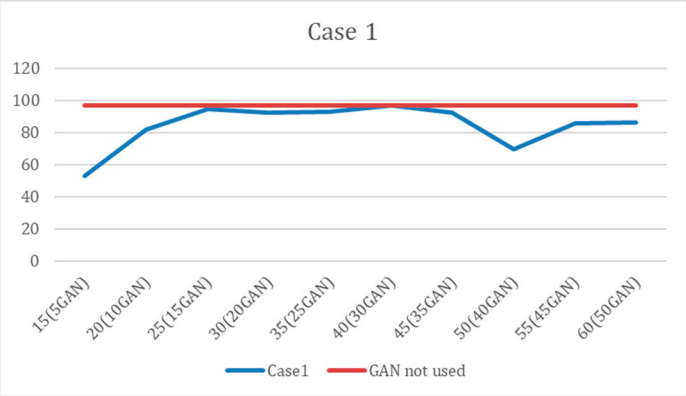


Figure 3. Accuracy comparison chart for Case 1.

4.4. Amplification Case Analysis

Following the analysis of Case 1, the study expanded the experiment to include all five cases. After completing the analysis for Case 1, all GAN samples were removed, and Case 2 was analyzed using the same method (Table 6), followed by Cases 3, 4, and 5. The same process of reducing the sample size and generating GAN samples was repeated for each case, with 50 training sessions conducted for each combination of original and generated samples.

Table 6. Number of training and testing samples for Case 2.

Datasets	Number of Training Samples	Number of Testing Samples
Case 1	50	10
Case 2	10 + 60 GAN	10
Case 3	50	10
Case 4	50	10
Case 5	50	10

Note: The CNN was originally designed with seven output nodes to accommodate potential future expansions. For this study, the architecture was adjusted to five output nodes to match the dataset, which included five speaker classes (four male and one female).

The results for all cases are presented in Table 7 and Figure 4. These results confirm that the improvement in accuracy due to GAN generation is more significant for cases with initially low accuracy. For cases where the initial accuracy was already high, GAN-generated samples offered little improvement. It is important to note that 'Cases' in Table 1 refer to speaker classes, while 'Cases' in Table 7 describe experimental configurations involving different ratios of GAN-generated to real samples.

Table 7. Accuracy of generated training for each case. In this table, 'Cases' refer to experimental configurations where the proportion of GAN-generated samples to real samples is systematically varied.

Number of Samples	Case 1	Case 2	Case 3	Case 4	Case 5
10 (initial data)	96.6%	99.6%	72.6%	71.8%	96.5%
25 (15 GAN)	52.8%	95%	42.8%	39.2%	48.4%
30 (20 GAN)	81.6%	98.2%	56.6%	58.2%	75.8%
35 (25 GAN)	94.6%	97.4%	69.8%	78%	83.8%
40 (30 GAN)	92.2%	99.6% (0%)	65.8%	66.8%	99% (2.5%)
45 (35 GAN)	93%	99.4%	72.4%	61%	98.8%
50 (40 GAN)	96.6% (0%)	98%	76.8% (4.2%)	81.2% (9.4%)	93.6%
55 (45 GAN)	92.2%	99%	68.4%	72.8%	98.6%
60 (50 GAN)	69.4%	96.4%	56.8%	47.8%	66.4%
65 (55 GAN)	86%	98.2%	68%	70.6%	84.2%
70 (60 GAN)	86.4%	98.8%	72.2%	70.2%	91.8%

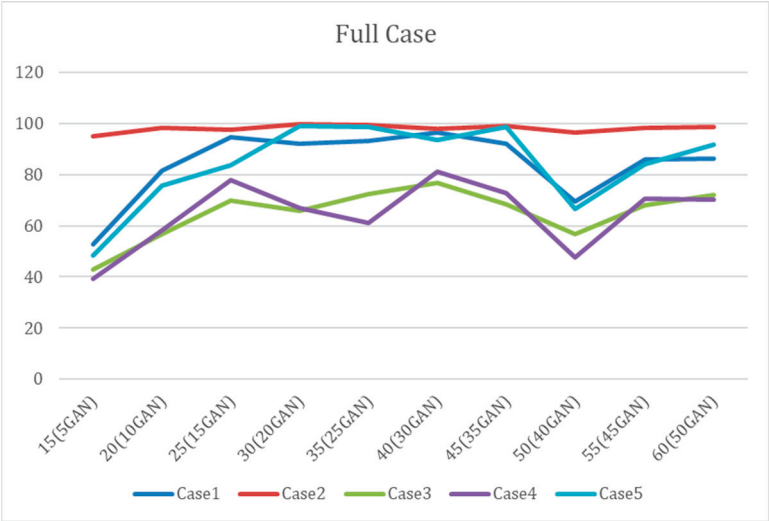


Figure 4. Trend chart for all dataset accuracies.

The "Cases" presented in Table 8 correspond to experimental configurations designed to evaluate the impact of different GAN-to-real sample ratios and dataset sizes on the performance of the proposed voiceprint recognition model. Each case represents a unique combination of real and GAN-generated samples, as outlined below:

Baseline Case (Case 1): This case uses only real samples (60 per speaker) without any data augmentation, serving as the baseline for comparison.

Low GAN-to-Real Ratio (Cases 2–5): These cases incorporate increasing numbers of GAN-generated samples, ranging from 20% to 50% of the real dataset size. For example, Case 2 includes 12 GAN-generated samples per speaker (20%), while Case 5 includes 30 samples per speaker (50%).

High GAN-to-Real Ratio (Cases 6–9): These cases examine configurations with higher GAN-to-real ratios, with GAN samples constituting 60% to 90% of the dataset. For instance, Case 6 includes 36 GAN samples per speaker (60%), while Case 9 uses 54 GAN samples (90%).

Fully Augmented Dataset (Case 10): This case consists of 100% GAN-generated samples (60 samples per speaker) to evaluate the model's performance when relying solely on synthetic data.

For all cases, the total number of samples per speaker was kept consistent at 100 to isolate the effects of the GAN-to-real sample ratio. The network configuration, including hyperparameters such as learning rate, batch size, and the number of training epochs, remained the same across all cases to ensure a fair comparison.

Table 8. Accuracy table over Case 1 to Case 10. In this table, 'Cases' represent experimental configurations with varying proportions of GAN-generated samples to real samples.

Case	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
10 (initial data)	79.4%	97%	69.8%	55.2%	83%	45.3%	96.4%	70%	74.3%	93%
25 (10 + 15)	78.8%	95.2%	74.2%	63.6%	87.4%	50.8%	97.4%	74.8%	83.4%	95.4% (2.4%)
30 (10 + 20)	81.4% (2%)	97.6%	79.8% (10%)	60.4%	89.2%	48.6%	96.4%	74%	81.6%	93.4%
35 (10 + 25)	58.6%	97%	66.6%	52.8%	75.6%	52%	95.4%	75.4%	83.6%	86.4%
40 (10 + 30)	74.2%	94%	69.4%	65.4%	79.8%	45.4%	84%	70.2%	58%	89.8%
45 (10 + 35)	66%	98.6% (1.6%)	68.8%	64.6%	90% (7%)	50.4%	95.2%	70%	72%	92.2%
50 (10 + 40)	76.2%	96.2%	69.6%	60.6%	87.6%	52.8%	90.2%	74.2%	77%	91.2%
55 (10 + 45)	75.4%	94%	77%	58.2%	84.8%	54.8% (9.5%)	95.8%	71.8%	88.8% (15.5%)	95.2%
60 (10 + 50)	74.6%	98.4%	76.6%	65.6% (10.4%)	82.8%	50.6%	98% (1.6%)	76.6% (6.6%)	80%	92.8%
65 (10 + 55)	65.2%	95%	45.2%	48.6%	65.2%	50.2%	97%	73.4%	79.4%	81.6%
70 (10 + 60)	64.2%	96.4%	71.2%	57%	83.6%	50%	97.4%	74%	85.8%	89.8%

4.5. Decreasing Analysis

In cases where the improvement in accuracy after generating GAN samples was less than 5%, the study reduced the number of original samples. This reduction helped investigate whether the original high accuracy was the reason GAN-generated samples had little effect. Reducing the number of initial samples lowered the overall accuracy, which allowed the study to evaluate whether the GAN's contribution was more noticeable with fewer original samples.

Table 9 shows the results for cases with an improvement of less than 5%. In these cases, reducing the number of original samples significantly increased the contribution of GAN-generated samples, confirming that the high initial accuracy limited the effect of GAN generation in previous tests.

Table 9. Accuracy table for Cases 1, 2, 7, and 10.

Number of Samples	Case 1	Case 2	Case 7	Case 10
5 (initial data)	47.8%	89.4%	91.2%	63.8%
20 (15 GAN)	56.2% (8.4%)	87.6%	85%	70.6%
25 (20 GAN)	63.4%	87.8%	88.2%	61.4%
30 (25 GAN)	64.2%	88.6%	92%	72.4% (8.6%)
35 (30 GAN)	46.4%	90%	91.2%	51.8%
40 (35 GAN)	53.8%	93%	97.2% (6%)	61.2%
45 (40 GAN)	63%	91.8%	88%	71.8%
50 (45 GAN)	51.4%	87.4%	86.8%	69.4%
55 (50 GAN)	59.6%	94.6% (5.4%)	92%	68.8%
60 (55 GAN)	61.8%	92.8%	91.6%	67.6%
65 (60 GAN)	47.4%	88.4%	87.8%	40.6%

4.6. Analysis Conclusions

The experiments verified that the use of GAN-generated samples significantly improves the performance of the CNN-MFCC voiceprint recognition model in cases with limited data, where the original accuracy is low. The number of GAN samples generated is also critical: blindly increasing the number of generated samples does not always lead to higher accuracy. Instead, selecting the right number of GAN-generated samples for each case can yield better results. Figure 5 illustrates the trend of accuracy (%) as the number

of GAN-generated samples increases. Accuracy improves significantly up to 40 samples but declines beyond this point, indicating a potential overfitting effect with excessive synthetic data. This visualization complements the experimental results by showing how accuracy initially improves but later declines due to potential overfitting when the number of GAN-generated samples becomes excessive. The study evaluated various GAN-to-real sample mixing ratios to identify optimal configurations for practical applications. In resource-limited scenarios, a higher proportion of GAN-generated samples (e.g., 60–70%) is beneficial for enriching diversity, whereas a balanced ratio (30–50%) is preferable in data-rich environments to ensure robustness without overfitting.

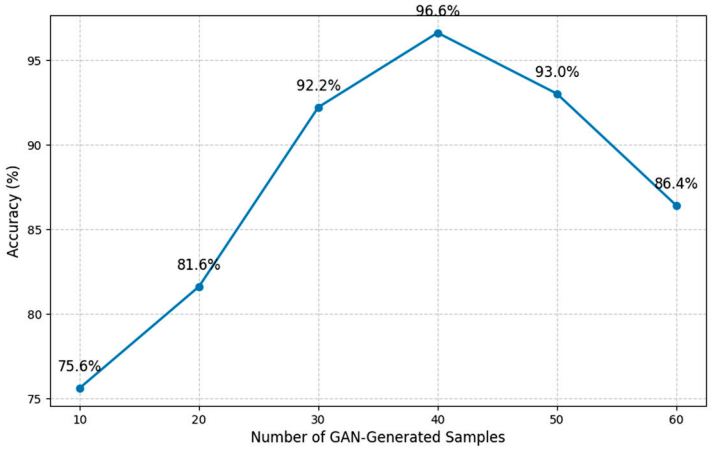


Figure 5. Trend of accuracy with increasing numbers of GAN-generated samples.

5. Discussion

While GANs, CNNs, and MFCCs have been extensively studied as standalone components, their integration in this work addresses a pressing practical challenge in voiceprint recognition: mitigating overfitting and improving generalization in limited-data scenarios. The iterative reduction technique, which adjusts the balance between real and GAN-generated samples, provides a structured methodology to optimize model performance. This approach is particularly relevant for applications requiring robustness across diverse speakers and environments, where traditional data augmentation methods may fail to generalize. Our findings highlight the potential of this technique to enhance data diversity without compromising model stability, offering a reproducible framework for practitioners facing similar data limitations.

This research introduces a novel method for improving the precision of voiceprint recognition by utilizing GANs to generate synthetic audio samples. Integrating these GAN-produced samples with high-quality data effectively conquers the challenges posed by limited dataset size and diversity, which are crucial for effective voice recognition systems. The proposed GAN-enhanced CNN approach is well-suited for practical applications such as multilingual access control systems, automated customer service, and forensic speaker identification. Future research could explore advanced GAN architectures like StyleGAN for more realistic audio synthesis and adaptive sampling methods to optimize model performance across diverse datasets.

Our study successfully executed a hybrid data augmentation technique that maintains a careful equilibrium between synthetic and natural samples. This equilibrium notably enhanced the model's ability to generalize across various voice samples, resulting in a robust method. Our tests revealed that incorporating GAN-generated samples enabled the model to reach higher accuracy, especially in scenarios with initially sparse real data, underscoring the importance of synthetic data for expanding voiceprint datasets without incurring excessive resource costs.

Further analysis revealed that, although GAN-generated data significantly boosted accuracy, their effectiveness varied based on context. When the initial accuracy was already

high, adding synthetic data offered limited additional benefits, suggesting there is a point at which more data do not help much in data-rich situations. In situations where the baseline accuracy was low, adding synthetic samples led to considerable improvements, showcasing the effectiveness of GANs in environments with limited resources. The research also highlighted the necessity of controlling the volume of synthetic data to prevent overfitting, a situation where the model becomes too specialized to the training data and performs poorly on new, unseen data, identifying an ideal range that enhances accuracy while allowing the model to generalize well.

The approach and results of this research offer a hopeful framework for voiceprint recognition technologies. These technologies can gain advantages from flexible and resource-efficient methods of data augmentation. Future research opportunities include enhancing GAN architectures to produce even more lifelike audio and investigating adaptive sampling methods, such as stratified sampling or active learning, to improve model performance across various voiceprint situations.

Author Contributions: Conceptualization, M.-L.H. and H.-Y.C.; methodology, Y.-S.L. and H.-Y.C.; software, T.-Y.H.; validation, Y.-S.L. and T.-Y.H.; writing—original draft preparation, Y.-S.L. and H.-Y.C.; writing—review and editing, M.-L.H. and H.-Y.C.; supervision, M.-L.H. and Y.-S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science and Technology Council and National Chin-Yi University of Technology, with fund number: MOST 111-2221-E-167-007-MY3, NSTC 112-2221-E-167-032-MY2, and NCUT 25-R-MI-027.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the funding protocol of National Science and Technology Council.

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

References

1. Wan, L.; Wang, Q.; Papir, A.; Moreno, I.L. Generalized end-to-end loss for speaker verification. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 4879–4883.
2. Kang, Y.; Cai, Z.; Tan, C.-W.; Huang, Q.; Liu, H. Natural language processing (NLP) in management research: A literature review. *J. Manag. Anal.* **2020**, *7*, 139–172. [CrossRef]
3. Snyder, D.; Garcia-Romero, D.; Povey, D.; Khudanpur, S. Deep neural network embeddings for text-independent speaker verification. In Proceedings of the Interspeech, Stockholm, Sweden, 20–24 August 2017; pp. 999–1003.
4. Chung, J.S.; Nagrani, A.; Zisserman, A. Voxceleb2: Deep speaker recognition. *arXiv* **2018**, arXiv:1806.05622.
5. Lei, Y.; Scheffer, N.; Ferrer, L.; McLaren, M. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 1695–1699.
6. Ravanelli, M.; Bengio, Y. Speaker recognition from raw waveform with sincnet. In Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 18–21 December 2018; pp. 1021–1028.
7. Heigold, G.; Moreno, I.; Bengio, S.; Shazeer, N. End-to-end text-dependent speaker verification. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 5115–5119.
8. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
9. Davis, S.; Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366. [CrossRef]
10. Rabiner, L.R.; Juang, B.-H. *Fundamentals of Speech Recognition*; Tsinghua University Press: Beijing, China, 1999.
11. Muda, L.; Begam, M.; Elamvazuthi, I. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *arXiv* **2010**, arXiv:1003.4083.
12. Logan, B. Mel frequency cepstral coefficients for music modeling. In Proceedings of the ISMIR, Plymouth, MA, USA, 23–25 October 2000; p. 11.
13. Kim, C.; Stern, R.M. Power-normalized cepstral coefficients (PNCC) for robust speech recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 1315–1329. [CrossRef]

14. Wang, C.; Zhang, M.; Xu, Z.-W.; Chen, C.; Sheng, D.-S. Effects of anisotropic ionospheric irregularities on space-borne SAR imaging. *IEEE Trans. Antennas Propag.* **2014**, *62*, 4664–4673. [CrossRef]
15. Alkhatib, B.; Eddin, M.M.K.; Syria, D. ASR Features Extraction Using MFCC And LPC: A Comparative Study. *J. Digit. Inf. Manag.* **2023**, *21*, 39. [CrossRef]
16. Saito, Y.; Takamichi, S.; Saruwatari, H. Statistical parametric speech synthesis incorporating generative adversarial networks. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2017**, *26*, 84–96. [CrossRef]
17. Wang, S.; Yang, Y.; Wu, Z.; Qian, Y.; Yu, K. Data augmentation using deep generative models for embedding based speaker recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 2598–2609. [CrossRef]
18. Pascual, S.; Bonafonte, A.; Serra, J. SEGAN: Speech enhancement generative adversarial network. *arXiv* **2017**, arXiv:1703.09452.
19. Sisman, B.; Yamagishi, J.; King, S.; Li, H. An overview of voice conversion and its challenges: From statistical modeling to deep learning. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *29*, 132–157. [CrossRef]
20. Kaneko, T.; Kameoka, H.; Hiramatsu, K.; Kashino, K. Sequence-to-Sequence Voice Conversion with Similarity Metric Learned Using Generative Adversarial Networks. In Proceedings of the Interspeech, Stockholm, Sweden, 20–24 August 2017; pp. 1283–1287.
21. Chandna, P.; Miron, M.; Janer, J.; Gómez, E. Monoaural audio source separation using deep convolutional neural networks. In Proceedings of the Latent Variable Analysis and Signal Separation: 13th International Conference, LVA/ICA 2017, Grenoble, France, 21–23 February 2017; Proceedings 13. pp. 258–266.
22. Skerry-Ryan, R.; Battenberg, E.; Xiao, Y.; Wang, Y.; Stanton, D.; Shor, J.; Weiss, R.; Clark, R.; Saurous, R.A. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In Proceedings of the International Conference on Machine Learning, Stockholm Sweden, 10–15 July 2018; pp. 4693–4702.
23. Pan, Q.; Pan, Y.; Zhou, J.; Wang, H.; Tao, L.; Kwan, H.K. Cyclegan with dual adversarial loss for bone-conducted speech enhancement. In Proceedings of the TENCON 2022—2022 IEEE Region 10 Conference (TENCON), Hong Kong, 1–4 November 2022; pp. 1–4.
24. Campbell, W.M.; Sturim, D.E.; Reynolds, D.A. Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Process. Lett.* **2006**, *13*, 308–311. [CrossRef]
25. Reynolds, D.A.; Quatieri, T.F.; Dunn, R.B. Speaker verification using adapted Gaussian mixture models. *Digit. Signal Process.* **2000**, *10*, 19–41. [CrossRef]
26. Bimbot, F.; Bonastre, J.-F.; Fredouille, C.; Gravier, G.; Magrin-Chagnolleau, I.; Meignier, S.; Merlin, T.; Ortega-García, J.; Petrovska-Delacrétaz, D.; Reynolds, D.A. A tutorial on text-independent speaker verification. *EURASIP J. Adv. Signal Process.* **2004**, *2004*, 101962. [CrossRef]
27. Kinnunen, T.; Li, H. An overview of text-independent speaker recognition: From features to supervectors. *Speech Commun.* **2010**, *52*, 12–40. [CrossRef]
28. Mak, M.-W.; Yu, H.-B. A study of voice activity detection techniques for NIST speaker recognition evaluations. *Comput. Speech Lang.* **2014**, *28*, 295–313. [CrossRef]
29. Burget, L.; Matejka, P.; Schwarz, P.; Glembek, O.; Cernocky, J.H. Analysis of feature extraction and channel compensation in a GMM speaker recognition system. *IEEE Trans. Audio Speech Lang. Process.* **2007**, *15*, 1979–1986. [CrossRef]
30. Mozilla. Common Voice. Available online: https://commonvoice.mozilla.org/en/datasets (accessed on 2 May 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Estimating the Spectral Response of Eight-Band MSFA One-Shot Cameras Using Deep Learning

Pierre Gouton [1,*], Kacoutchy Jean Ayikpa [1,2] and Diarra Mamadou [1]

[1] Laboratoire Imagerie et Vision Artificielle (ImVia), Université de Bourgogne, 21000 Dijon, France; kacoutchy-jean.ayikpa@u-bourgogne.fr or kacoutchy.ayikpa@uvci.edu.ci (K.J.A.); patoudiarra@gmail.com (D.M.)

[2] Unité de Recherche et d'Expertise Numérique (UREN), Université Virtuelle de Côte d'Ivoire, 28 BP 536, Abidjan 28, Côte d'Ivoire

* Correspondence: pgouton@u-bourgogne.fr; Tel.: +33-782194200

Abstract: Eight-band one-shot MSFA (multispectral filter array) cameras are innovative technologies used to capture multispectral images by capturing multiple spectral bands simultaneously. They thus make it possible to collect detailed information on the spectral properties of the observed scenes economically. These cameras are widely used for object detection, material analysis, and agronomy. The evolution of one-shot MSFA cameras from 8 to 32 bands makes obtaining much more detailed spectral data possible, which is crucial for applications requiring delicate and precise analysis of the spectral properties of the observed scenes. Our study aims to develop models based on deep learning to estimate the spectral response of this type of camera and provide images close to the spectral properties of objects. First, we prepare our experiment data by projecting them to reflect the characteristics of our camera. Next, we harness the power of deep super-resolution neural networks, such as very deep super-resolution (VDSR), Laplacian pyramid super-resolution networks (LapSRN), and deeply recursive convolutional networks (DRCN), which we adapt to approximate the spectral response. These models learn the complex relationship between 8-band multispectral data from the camera and 31-band multispectral data from the multi-object database, enabling accurate and efficient conversion. Finally, we evaluate the images' quality using metrics such as loss function, PSNR, and SSIM. The model evaluation revealed that DRCN outperforms others in crucial performance. DRCN achieved the lowest loss with 0.0047 and stood out in image quality metrics, with a PSNR of 25.5059, SSIM of 0.8355, and SAM of 0.13215, indicating better preservation of details and textures. Additionally, DRCN showed the lowest RMSE 0.05849 and MAE 0.0415 values, confirming its ability to minimize reconstruction errors more effectively than VDSR and LapSRN.

Keywords: spectral response; MSFA one-shot; deep learning; multispectral image

1. Introduction

Acquiring high-spectral-resolution multispectral images from low-resolution sensors represents a significant challenge in various fields, including remote sensing, medical imaging, precision agriculture, and scene analysis [1]. High-resolution multispectral images provide a more detailed and accurate analysis of observed scenes, which is important for environmental monitoring typo and disease-detection applications [2]. The acquisition of multispectral images through systems based on CRI, Hyspex, etc., provides images of 30 to 100 bands, offering unparalleled information [3]. Each spectral band makes it possible to detect specific characteristics of objects that are not visible to the naked eye. As in precision agriculture, these images can help identify areas requiring particular treatments. They can detect tissue abnormalities in healthcare and find applications in many areas. However, its devices are not fast and poorly suited to slow-motion objects. 8-band one-shot MSFA (multispectral filter array) cameras provide a compact, cost-effective solution for spectral data acquisition. However, they produce images of low spectral resolution. This band

limitation makes capturing the details needed for in-depth analyses difficult. Therefore, although these sensors are accessible, the quality of the data they provide is insufficient for many critical applications. Thus, as part of the European CAVIAR project, which aims to improve image-detection technologies in various sectors, notably health, agriculture, etc., by relying on advanced imaging techniques, the use of an 8-band MSFA filter was used for the camera setup to capture a broader spectrum of light [4,5]. However, the transition to a high-spectral-resolution system (31 to 100 bands) requires more sophisticated and expensive sensors, which are slow, limiting their accessibility. This is why estimating the spectral response of 8-band one-shot MSFA cameras raises significant technical challenges. In this context, deep learning methods are a promising solution. This study focuses on applying advanced deep learning techniques to improve the quality and accuracy of the spectral response of 8-band one-shot cameras to approximate hyperspectral systems. By harnessing the power of convolutional neural networks (CNN) and other deep neural-network architectures, it is possible to transform data captured by low-resolution sensors into high-quality multispectral images [6]. This study aims to demonstrate the potential of deep learning techniques to approximate the spectral response of 8-band one-shot cameras, making this technology more accessible and applicable to a wide range of domains. The results obtained open new perspectives for improving sensor systems and image analysis methods, thus contributing to significant advances in research and practical applications.

The specific contributions of our study are as follows:

- We have implemented image datasets that contain 8-band images as input and 31-band images as output. We retrieved the images from the TokyoTech [7] database, which offers 31-band multispectral images. Using a Gaussian projection, we generated 8-band images from this data. These images are then mosaicked and demosaicked to achieve a quality comparable to the camera developed as part of the CAVIAR project. This approach creates a robust dataset for training and evaluating models for estimating the spectral response of 8-band one-shot cameras.
- We have developed several models based on deep-learning algorithms, including very deep super-resolution (VDSR), deeply recursive convolutional networks (DRCN), and Laplacian pyramid super-resolution networks (LapSRN). These models were initially used for super-resolution and then adapted to approximate the spectral response. These models are powerful tools for approximating the spectral response of 8-band one-shot MSFA cameras. By exploiting the advanced architectures of these networks, we aim to significantly improve the quality of the spectral response obtained with these cameras.
- We compare the algorithms used for building the models by evaluating them using specific metrics such as structural similarity index (SSIM) and peak signal-to-noise ratio (PSNR) to measure their performance. Furthermore, a visual comparison between the predicted spectral responses and the ground truths is performed to validate the effectiveness of the proposed methods. This approach allows us to identify the best-performing models and optimize techniques for approximating the spectral response of 8-band one-shot MSFA cameras.

The rest of our manuscript is organized as follows: Section 2 deals with the state of the art regarding the spectral response of multispectral cameras. Section 3 details the materials and methods used in our approach. Section 4 presents the experiments carried out and the results obtained. Section 5 provides a discussion of these results. Section 6 addresses the threats to validity. Finally, Section 7 presents our conclusion.

2. Related Work

Several works have been carried out in the reconstruction of images, and we can cite, among others, Toivonen et al., who developed a practical method for estimating the spectral response of camera sensors and the associated uncertainty using a specific imaging process. This method relies on a restricted set of 15 images, including diffraction images and color patches of known spectra, to obtain high-resolution spectral response

estimates. The approach is versatile and applicable to any camera sensor in the visible domain. It does not impose strict constraints on the possible estimates, thus ensuring high precision of the results, which agree with previous estimates [8]. Darrodi et al. conducted a study to test the performance of different sensor estimation techniques from the literature. They used measured and synthetic data, which they then compared to reference data to evaluate the effectiveness of these techniques on two cameras [9]. Han et al. proposed an estimation method based on a single image and exploiting fluorescence without requiring prior knowledge of the illumination spectrum. Under different light sources, the spectral distributions of the fluorescence emitted by the same material remain constant, up to one scale. Thus, the camera response to fluorescence maintains the same chromaticity. By relying on this chromatic invariance, they estimated the camera's spectral sensitivity under arbitrary illumination, the spectrum of which is unknown. Through numerous experiments, they demonstrated the accuracy of their method under various lighting conditions. Additionally, they showed how to recover daylight spectra from the estimated results and then use the cameras' estimated spectral sensitivities and daylight spectra to solve color correction problems [10]. Zhao et al. studied spectral features using singular value decomposition (SVD) to extract basis functions. They collected data from the literature and direct measurements of the sensitivity of different cameras. Their paper compares the extracted basis functions to identify the optimal set with other mathematical basis functions. These optimized basis functions can then be used to estimate the unknown spectral sensitivity of any camera [11]. Prasad et al. proposed a quick and straightforward method to obtain a reasonable approximation of camera-response functions using only a color chart and unknown, random lighting. This approach is made possible by carefully designing the cost function, which imposes several constraints to make the problem feasible. Among the components of this cost function, the Luther condition serves as a global shape prior, while commercial lighting systems with unknown spectra provide narrow spectral windows for local reconstruction. The quality of the rebuilding obtained is comparable to that of methods using known lighting [12]. Matanga et al. developed an innovative method to approximate the spectral response of color images. They introduced an approach based on neural networks, aiming to improve the two previous methods: those using circular and exponential functions and those based on the inverse of Penrose or Wiener. This new method can also be applied to calibrating most color scanning systems and subwavelength filters [13].

When we talk about a camera's spectral response, we also talk about the quality and reconstruction of the images. Chunwei et al. thus conducted a comparative study on GANs from different angles to address these issues. They began by reviewing recent advances in GANs and then presented popular architectures suitable for image super-resolution, both for small and large samples. They analyzed the motivations, implementations, and differences of optimization methods based on GANs and discriminative learning, including supervised, semi-supervised, and unsupervised approaches. In addition, a comparison of the performance of these GANs was carried out on public datasets, accompanied by a quantitative and qualitative analysis [14]. For their part, Vizilter et al. proposed a new approach to interconnect multispectral images via diffusive morphology techniques to improve aviation vision systems. Unlike traditional morphological methods, this approach does not require image segmenting to describe its shape and offers strong resistance to noise and distortions. They developed procedures for interconnecting two- and three-channel images using diffusive morphological filtrations and an image pyramid. A promising prototype of this improved vision system was tested with multispectral data collected during an aeronautical test [15]. Andriyanov et al. focused on the problem of filtering multispectral images by studying the effectiveness of filtering image sequences. They analyzed variations in error variance as a function of the number of images in the sequence and the inter-image correlation coefficient. To statistically model these images, they used doubly stochastic random fields [16]. Furthermore, Andriyanov et al. explored the restoration of images under conditions where a portion of the observations is subject to additive noise,

or more precisely, the restoration of randomly sampled images with pixels affected by white Gaussian noise. They developed nonlinear filters based on deep doubly stochastic Gaussian models, which were more effective than linear methods and classical algorithms. These filters allow images to be restored using only 50% of the data with a relative error rate of 9% [17]. These studies show the importance of robust and innovative approaches to improve image quality, particularly in multispectral vision and image reconstruction, where precision and resistance to noise are essential.

Most existing methods for approximating spectral responses must be adapted to our specific problem of 8-band MSFA one-shot cameras. Our article presents a new approach based on deep learning to address this challenge.

3. Materials and Methods

The experiments were conducted using Python programming at the ImViA laboratory on a DELL desktop computer. This system has an Intel(R) Core i7-10700 CPU running at 2.90 GHz, with 32 GB of RAM and an NVIDIA Quadro P400 GPU. The models were configured in Python version 3.8.8, utilizing Keras API version 2.4.3 and TensorFlow version 2.3 as the backend, with CUDA 10.1 and cuDNN 7.6 dependencies for GPU acceleration.

3.1. Dataset

Our dataset comes from the multispectral database called TokyoTech. This dataset contains 31-band multispectral images from 420 nm to 720 nm at 10 nm intervals. Images were captured using a monochrome camera with a VariSpec liquid crystal tunable filter (VariSpec VIS). This dataset is composed of 35 different objects, each containing 31 spectral bands [7].

Figure 1 shows an extract of some images from the database.

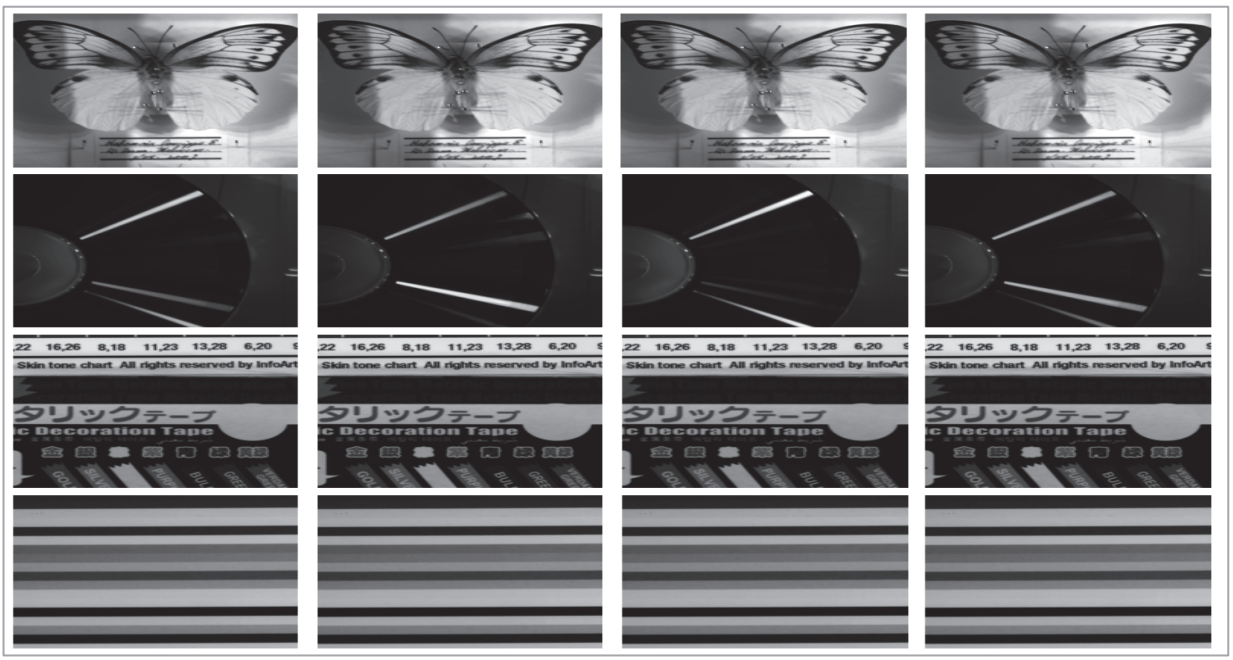


Figure 1. Extracted from the TokyoTech database [7].

We divided each image into 4 segments from this dataset to create a more extensive database for our experiment. This will allow us to obtain a dataset of 140 images containing 31 spectral bands. Figure 2 shows some images from the new dataset.

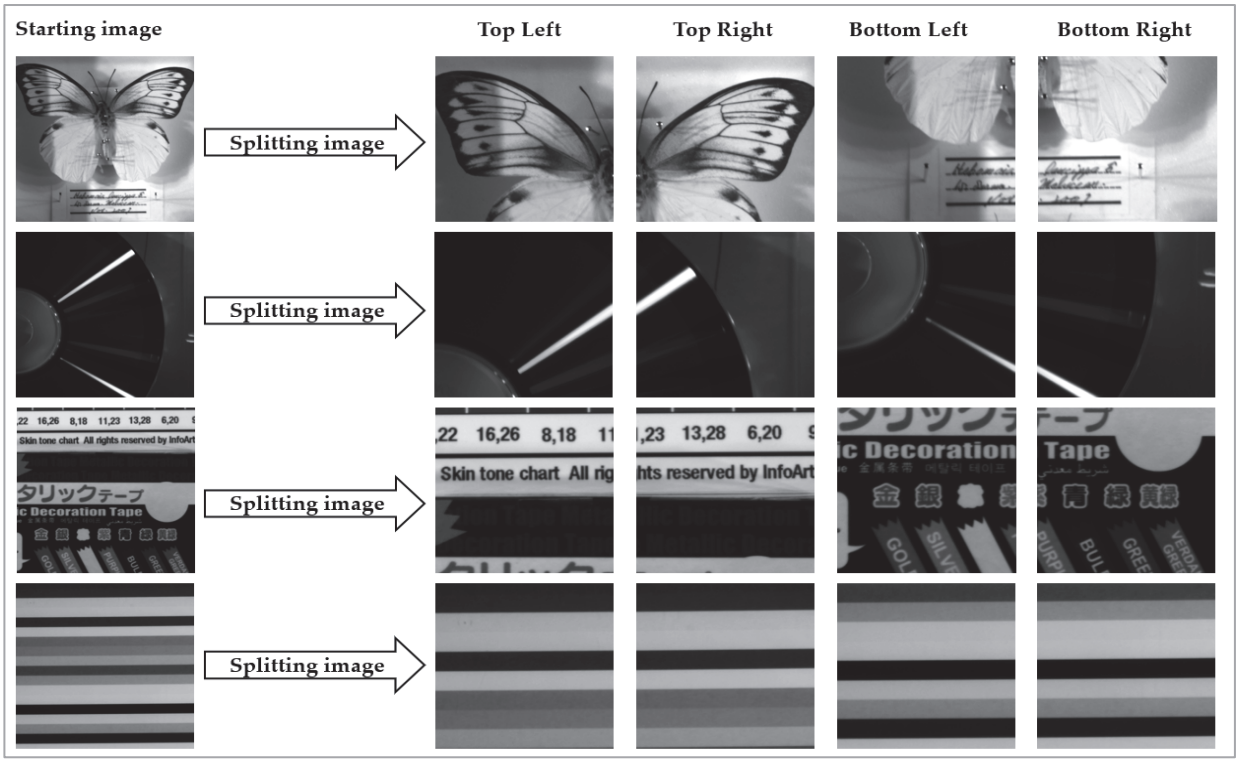


Figure 2. Illustration of split image.

After splitting the data, we used a Gaussian projection, an ideal method to model the MSFA filters of the color shade hybrid sensor and obtain spectral responses from 31 to 8 bands. This projection reduces the dimensionality of the data while retaining essential spectral characteristics, thus facilitating the precise estimation of spectral bands from a reduced number of initial bands. Figure 3 shows the transformation of the 31 bands into 8 bands.

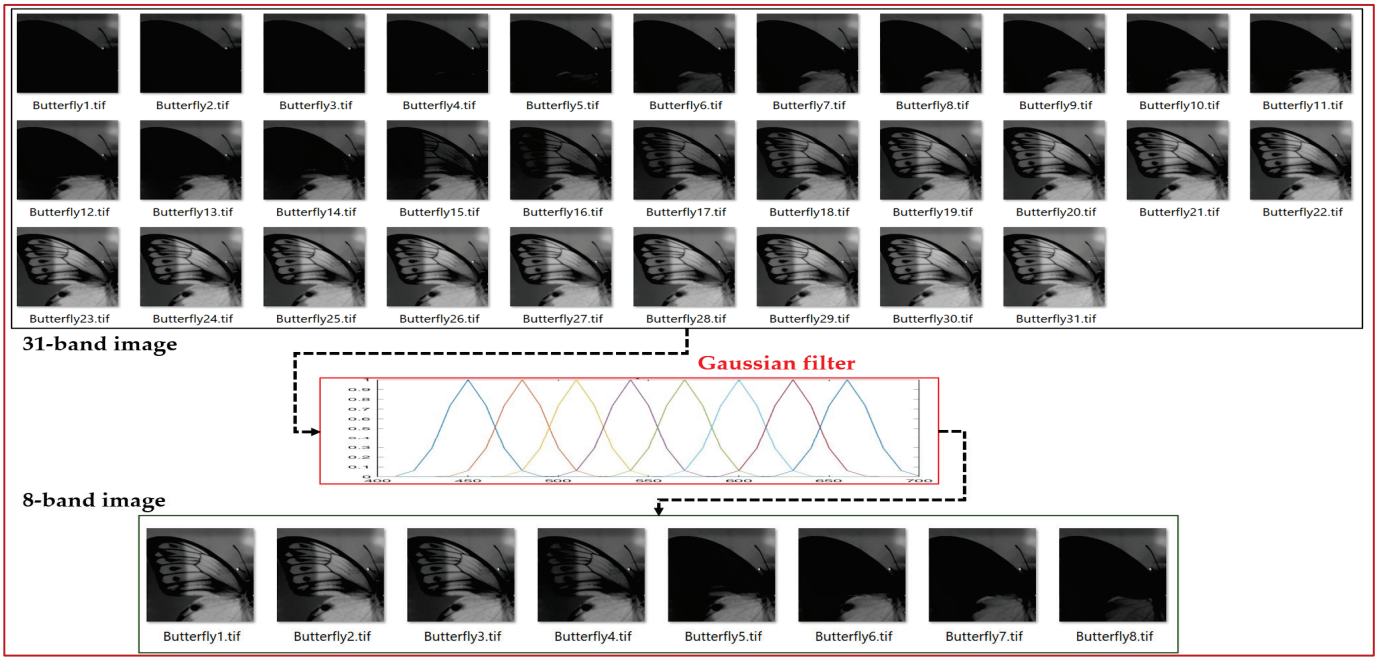


Figure 3. Illustration of the Gaussian projection of images from 31 bands to 8 bands.

Once the Gaussian projection is performed on the 31-band images to reduce them to 8 bands, we move on to the mosaic creation step. This mosaic results from assembling the 8 image strips to form a larger, more informative composite image. This mosaic is then demosaicked using the bilinear method. This bilinear method makes it possible to restore best the images captured by the camera developed as part of the CAVIAR project while preserving spectral details. Using bilinear demosaicking ensures a more faithful representation of the original data, which is essential for subsequent analyses and applications.

Figure 4 shows the images obtained before and after the operation.

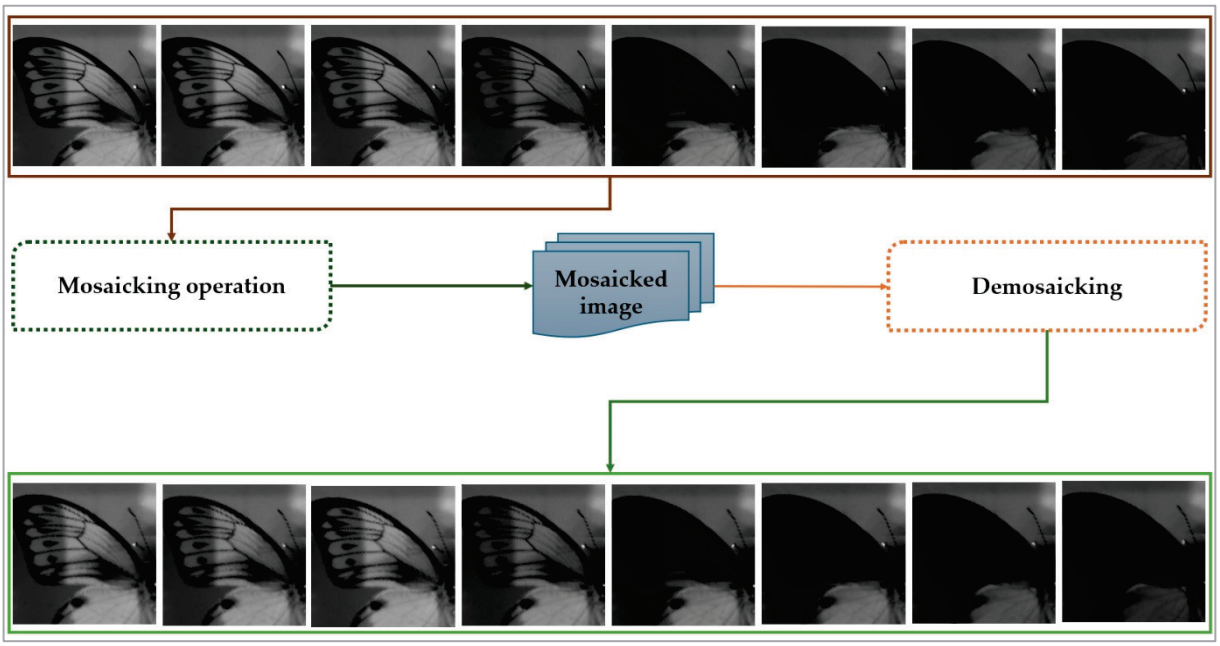


Figure 4. Illustration of the double operation of mosaicking and demosaicking of 8-band images.

Mosaicking is a fundamental process that creates a composite image from multiple-pixel images captured at different wavelengths. This process takes place in several key stages:

1. When capturing an image with our multispectral camera, several individual images are generated. In our case, we obtain eight distinct images, each representing pixels with similar spectral properties. Although they provide detailed information about the observed scene, these images remain incomplete due to their dispersion.
2. The captured pixels are grouped according to their spectral properties. This grouping is essential to reduce data redundancy and improve the clarity of analysis.
3. To reconstruct a fully defined composite image, we rely on our MSFA (multispectral filter array) model [18]. This powerful tool generates mosaics from scattered images while preserving spectral information.

Demosaicking, a crucial part of the process, separates the composite image obtained during mosaicking to find the dispersed images. For each sparse image, demosaicking uses interpolation techniques to estimate the missing pixel values. In our case, we employed bilinear or bicubic interpolation, which evaluates missing values based on neighboring pixels. Once the missing values have been interpolated, the different scattered images are merged to create the demosaicked image.

Figure 6 shows mosaic images from the dataset.

Figure 5 illustrates the mosaicking and demosaicking process.

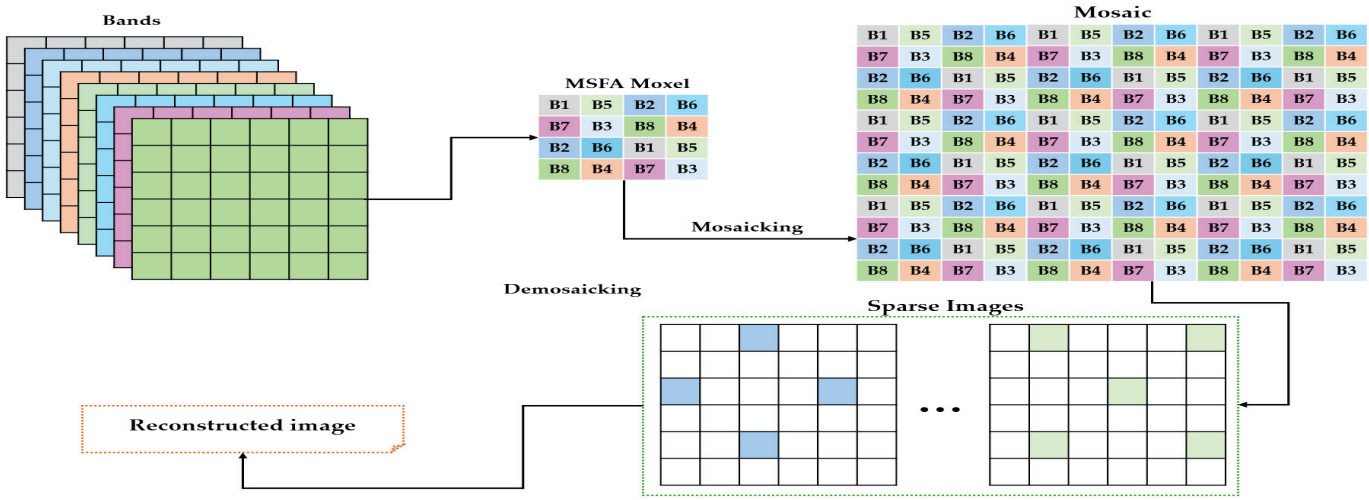


Figure 5. The mosaicking and demosaicking process.

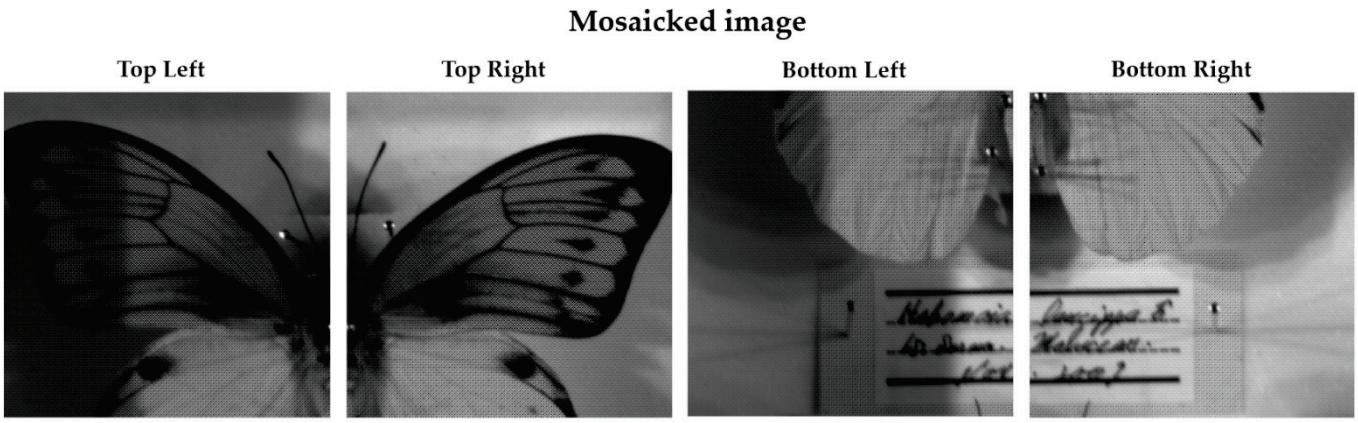


Figure 6. Presentation of mosaicked images.

Figure 7 summarizes the process of preparing our data for the experimentation of our study.

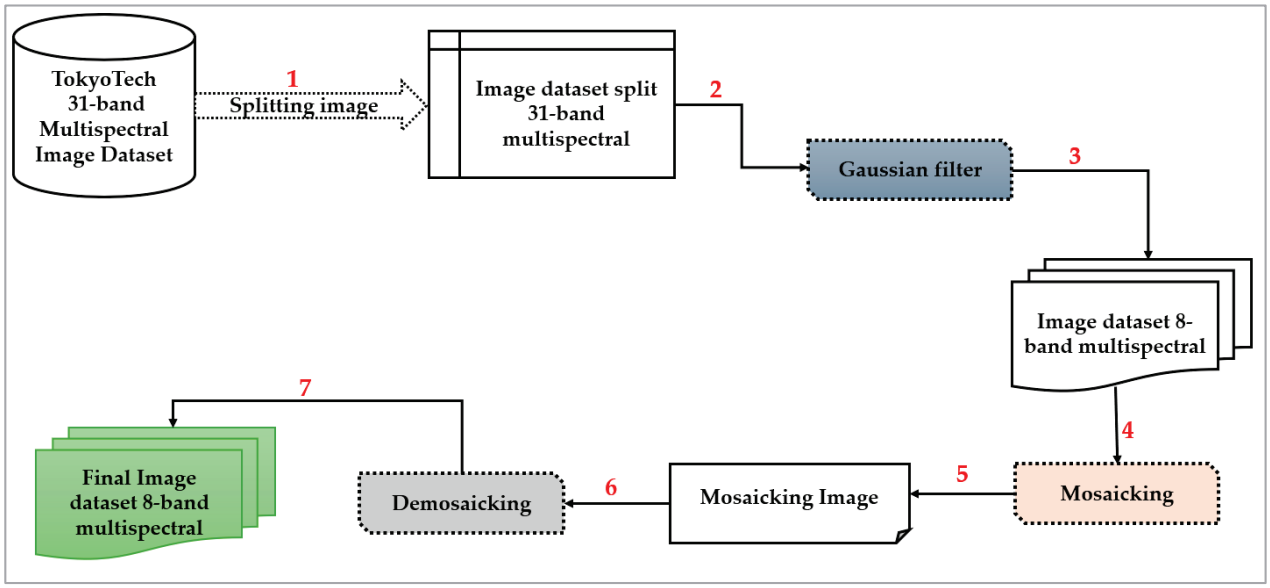


Figure 7. Summary of setting up the dataset.

3.2. Deep-Learning Algorithms

Image reconstruction is a growing field in image processing, aiming to improve image quality with finer and more precise details. Deep-learning algorithms have progressed considerably, enabling more accurate and detailed image reconstructions. Among these algorithms, several innovative approaches stand out for their ability to exploit deep and complex architectures to improve images [19].

In our study, we modify the architecture of three algorithms to meet our specific needs: very deep super-resolution (VDSR), Laplacian pyramid super-resolution network (LapSRN), and deeply recursive convolutional network (DRCN).

Each of these algorithms provides a unique method for tackling the reconstruction problem by exploiting their architectures' specific characteristics to optimize the reconstructed images' quality. When these algorithms process images, they must understand and model how each spectral band contributes to the final image. The spectral response of the cameras plays a role in this process, as it directly influences the quality of the input data for these algorithms. An accurate and well-calibrated spectral response allows algorithms to capture better and restore spectral details, thereby improving the quality and fidelity of reconstructed images.

3.2.1. Very Deep Super-Resolution (VDSR)

VDSR is a sophisticated deep-learning technique designed to improve image resolution. It uses a convolutional neural network (CNN) architecture of 20 layers, significantly more profound than traditional networks. This depth allows VDSR to effectively capture complex features and high-frequency details in images [20].

The architecture includes the following steps:

- **Input processing:** An 8-band multispectral image is fed to the first convolutional layer, which uses 64 filters of size 3×3 and applies a 'same' padding to maintain the dimensions of the input image. A function of ReLU activation is applied to introduce nonlinearity.
- **Hidden layers:** The image output from the first layer will be passed through 19 convolutional layers, followed by a final convolutional layer, which uses several filters equal to the number of output channels and applies 'same' padding. (20th layer). Each convolutional layer uses 64 filters of size 3×3 to extract features from the image.
- **Residual learning:** Another convolution is applied to the input image to bring it into the desired output shape. Residual addition combines the final layer's outputs and the input image's convolution. This will help the model learn the residual details needed to improve the image resolution.

The equation resulting from this model is the following:

- **I** is the input image.
- **K1, K2, . . ., Kn** is the convolution filters.
- Let **R** be the result of the last convolution layer.
- Let **O** be the output image.

$$X_i = ReLU(K_i * X_{i-1}) \; with \; X_0 = I \tag{1}$$

$$R = K_{final} * X_{18} \tag{2}$$

$$F = K_{residual} * I \tag{3}$$

$$O = R + F \tag{4}$$

Algorithm 1 presents the pseudocode of the VDSR algorithm.

Algorithm 1: VDSR

Input: I, K1, K2, . . ., Kn, K_{final}, $K_{residual}$
Output: O
Step 1: Initialize the input
$X_0 = I$
Step 2: Apply the sequence of convolutional layers
For i = 1 to 18:
$X_i = ReLU(K_i * X_{i-1})$
Step 3: Compute the result of the last convolutional layer
$R = K_{final} * X_{18}$
Step 4: Compute the residual connection
$F = K_{residual} * I$
Step 5: Calculate the final output image
$O = R + F$

Figure 8 shows the VDSR architecture implemented.

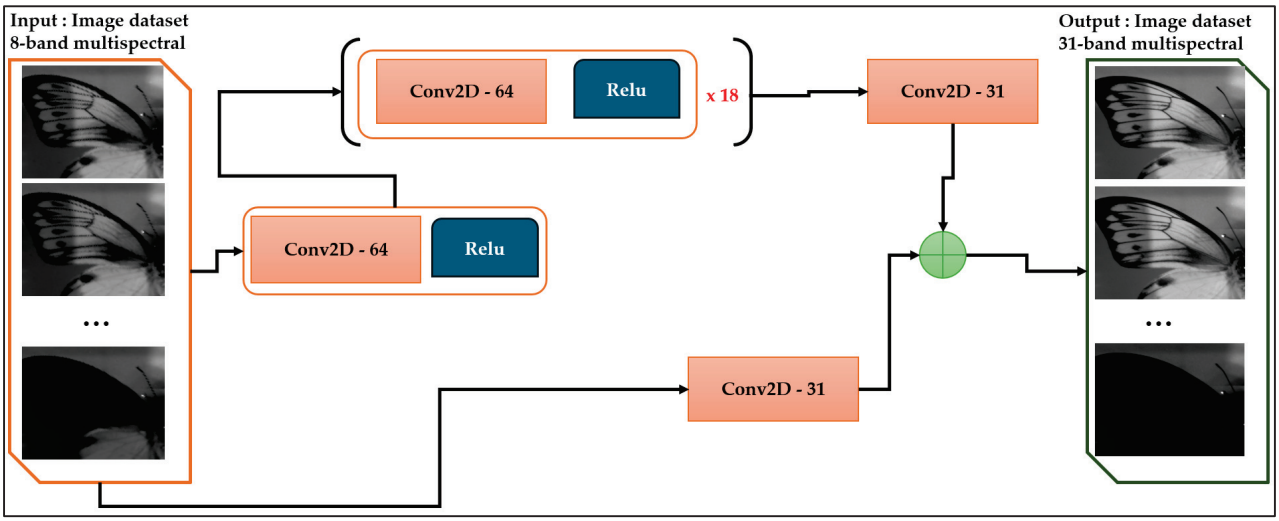


Figure 8. Modified VDSR architecture.

3.2.2. Deeply Recursive Convolutional Network (DRCN)

The deeply recursive convolutional network (DRCN) is an advanced convolutional neural-network architecture for image super-resolution. This model exploits deep recursion to improve the quality of images by increasing their resolution. DRCN is based on applying the same series of convolutional layers recursively, multiple times, to extract and refine image features gradually. This recursion allows the model to be deep and efficient regarding parameters [21]. Its architecture is as follows:

- **Input Layer:** The low-resolution image is first fed into a convolutional input layer, initialing feature extraction.
- **Recursive Convolutional Blocks:** The core of DRCN consists of several convolutional blocks that are applied recursively. Each block includes one or more convolutional layers followed by ReLU activation. The same set of layers is used multiple times to input the image.
- **Recursion:** Instead of directly stacking numerous convolutional layers, DRCN reuses the same layers to obtain an adequate depth without significantly increasing the parameters.
- **Fusion Layer:** Each recursive application's outputs are combined and fused to create a comprehensive, multi-scale image representation. This is a testament to DRCN's design's power to capture the full complexity of the input.
- **Reconstruction Layer:** Finally, a reconstruction layer takes the fused output and generates the final high-resolution image.

The equation resulting from this model is the following:

- **I** is the input image.
- **K** is the convolution filter for feature extraction.
- **K2, K3,. . ., Kn** are the recursive layers' convolution filters.
- $\mathbf{K_{final}}$ is the convolution filter for the reconstruction layer.
- **F** is the extracted feature maps.
- R_i is the result of the **i-th** recursive layer.
- **O** is the output image.

$$F = ReLU(K_1 * I) \tag{5}$$

$$R_i = ReLU(K_i * R_{i-1}) + F \; with \; R_0 = F \tag{6}$$

$$O = K_{final} * R_{16} \tag{7}$$

Algorithm 2 presents the pseudocode of the DRCN algorithm.

Algorithm 2: DRCN

Input: I, K1, K2, K3, . . ., Kn, K_{final}
Output: O
Step 1: Feature extraction using the first convolutional layer
$F = ReLU(K_1 * I)$
Step 2: Initialize the first recursive layer
$R_0 = F$
Step 3: Apply the recursive layers
For i = 1 to 16
$R_i = ReLU(K_i * R_{i-1}) + F$
Step 4: Reconstruction layer
$O = K_{final} * R_{16}$

Figure 9 shows the modified DRCN architecture implemented.

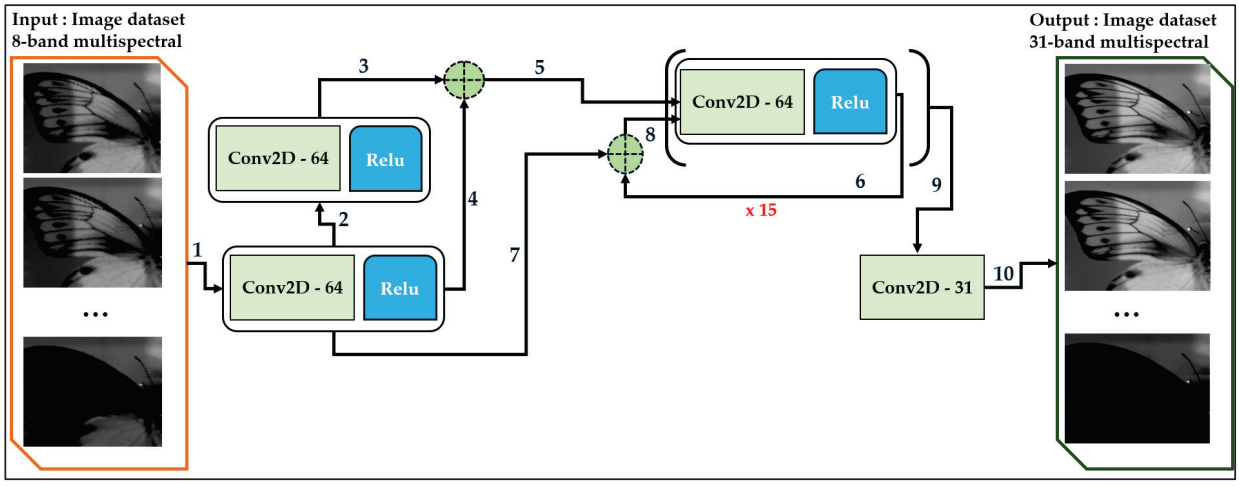


Figure 9. Modified DRCN architecture.

3.2.3. Laplacian Pyramid Super-Resolution Network (LapSRN)

The Laplacian pyramid super-resolution network (LapSRN) is a neural-network architecture designed for image super-resolution. It was introduced to improve image quality and resolution using a hierarchical approach inspired by the Laplacian pyramid.

LapSRN uses a Laplacian pyramid to represent images at different resolution scales. The Laplacian pyramid decomposes an image into a series of frequency subbands, allowing the network to process image details at various resolution levels stepwise [22,23].

- The LapSRN architecture includes several levels, each corresponding to a different resolution scale.
- Each pyramid level contains a convolutional block that extracts image features at that specific scale. This block consists of several convolutional layers followed by ReLU activation.

The equation resulting from this model is the following:

$$F = K_1 * I \tag{8}$$

- **F**: The initial output of the first transformation applied to the input image.
- K_1: A filter or convolution kernel applied to the input image.
- I: The input image could be a multispectral or low-resolution image.

This equation represents an initial transformation in which a convolution kernel K_1 is applied to the input image I to obtain the first representation F. This is used to extract low-level features, such as edges or textures.

$$L_j = K_{pyramid} * R_{10} \tag{9}$$

- L_j: An intermediate exit (perhaps at the pyramid level) at step j.
- $K_{pyramid}$: A kernel or filter linked to a pyramid-type structure, often used for multi-scale super-resolutions.
- R_{10}: The final output of the recursive step (probably after 10 iterations).

This equation suggests that the final recursion R_{10} is combined with a specific kernel $K_{pyramid}$, indicating a multi-scale mechanism. This allows information from different levels of detail to be aggregated, thereby enhancing the model's ability to produce images with fine features at different resolutions.

$$O = K_{final} * L_2 \tag{10}$$

- O: The final output of the model.
- K_{final}: A final kernel applied to the previous output for the final transformation.
- L_2: The output of the second stage in the pyramid or another network module.

This last equation combines the kernel K_{final} with the intermediate output L_2 to produce the result O. This final step can be seen as a refinement, where all the information obtained previously is integrated to generate the final version of the image with improved resolution or other desired properties.

Algorithm 3 presents the pseudocode of the LapSRN algorithm.

Algorithm 3: LapSRN

Input: I, K_1, K_{r1}, K_{r2}, $K_{pyramid}$, K_{final}
Output: O
Step 1: Apply the first convolution to extract features from the input image
$F = K_1 * I$
Step 2: Initialize the first residual block
$R_0 = F$
Step 3: Apply the residual blocks
For i = 1 to 10
$R_i = K_{r2} * ReLU(K_{r1} * R_{i-1}) + R_{i-1}$
Step 4: Create Laplace pyramid levels
For j = 1 to 2
$R_{10} = K_{pyramid} * R_{10}$
Step 5: Final output
$O = K_{final} * R_{10}$

Figure 10 shows the LapSRN architecture implemented.

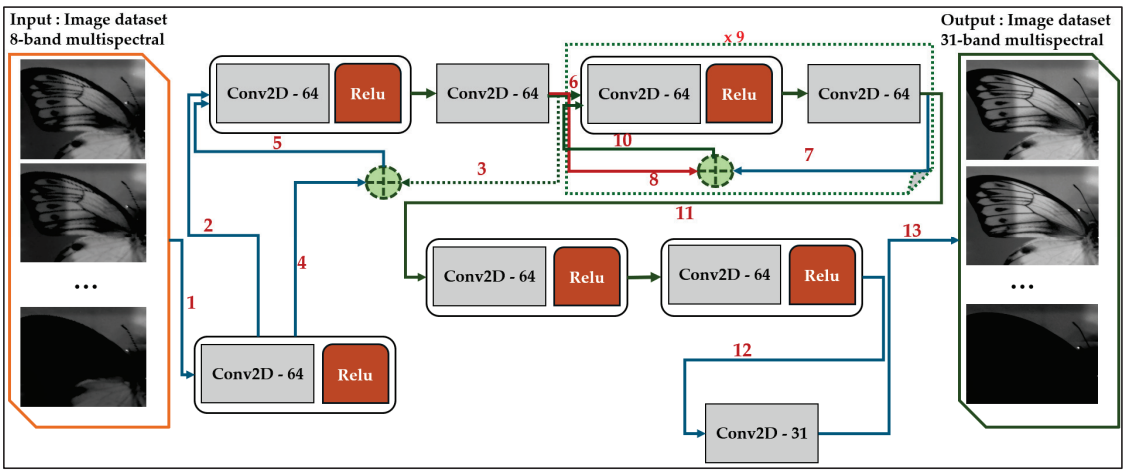


Figure 10. Modified LapSRN architecture.

3.3. The General Process of Our Method

Our general methodology is divided into three main parts:

First, we prepare the data necessary for our study. This step includes collecting, transforming, and normalizing the data to fit our model. We describe how we divide 31-band images into 8-band images using Gaussian projections. These transformed data are then organized into mosaics and demosaicked using a bilinear method, which makes it possible to simulate the images obtained by the camera developed as part of the CAVIAR project.

The second part of our methodology consists of implementing different models based on deep-learning algorithms. We use advanced architectures of algorithms that we have modified to adapt to our study, such as very deep super-resolution (VDSR), Laplacian pyramid super-resolution network (LapSRN), and deeply recursive convolutional network (DRCN). Each model is adjusted to optimize image reconstruction, exploiting the specific characteristics of their respective architectures to improve image resolution and quality. These models are trained with the prepared data, thereby learning to predict the spectral responses of the 31-band images from the 8-band images. Finally, we evaluate our models using test data. This step measures the performance and efficiency of the models. We use various metrics such as loss function, PSNR (peak signal-to-noise ratio), and SSIM (structural similarity index) to evaluate the quality of spectral responses. The evaluation allows us to identify the strengths and limitations of each model and make the necessary adjustments to improve their performance. Figure 11 illustrates the general outline of our methodology.

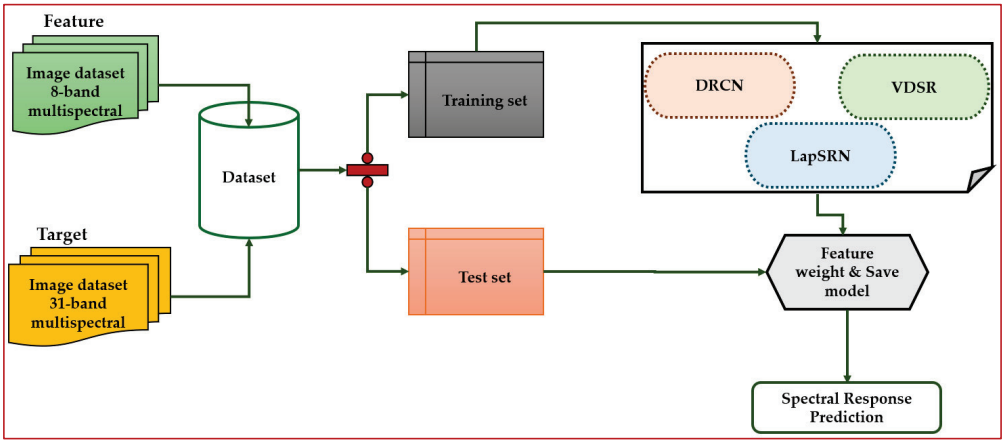


Figure 11. Block diagram of the methodology.

3.4. Metric Performance

To evaluate the performance of the spectral responses provided by our models, we use several metrics [18], including

- PSNR (peak signal-to-noise ratio): Peak signal-to-noise ratio (PSNR) is a commonly used metric to evaluate the quality of a reconstructed or compressed image compared to its original. It quantifies the ratio between the maximum possible power of a signal (the peak signal) and the power of the noise, which alters the fidelity of its representation.

$$PSNR = 10 \cdot \log_{10}\left(\frac{(2^n - 1)^2}{\frac{1}{n}\sum_{i=1}^{n}\left(I^i - \hat{I}^i\right)^2}\right) \tag{11}$$

- The spectral angle mapper (SAM) is a spectral classification method based on calculating the angle between two spectral vectors in a high-dimensional space. Its principle is based on the following elements: Each pixel in a hyperspectral image is represented by a spectral vector, where each dimension corresponds to the reflectance or luminance of the pixel in a specific spectral band. Each class or reference material is also represented by an average spectral vector calculated from selected reference pixels. The SAM calculates the angle between the pixel's spectral vector and each reference vector in n-dimensional space (where n corresponds to the number of spectral bands). The smaller the angle between two vectors, the more similar the spectra are. A maximum angular threshold determines whether a pixel should be assigned to a specific class [24].

$$SAM = \cos^{-1}\left(\frac{\sum_{i=1}^{n}\left(I^i \odot \hat{I}^i\right)}{\sqrt{\sum_{i=1}^{n}\left(I^{2^i} \odot \hat{I}^{2^i}\right)}}\right) \tag{12}$$

- SSIM (structural similarity index measure): SSIM is designed to assess the similarity between two images by comparing their structural information, luminance, and contrast while considering the characteristics of the human visual system. Unlike more straightforward measures such as mean square error (MSE) or peak signal-to-noise ratio (PSNR), which do not consider aspects of human perception, SSIM offers a more comprehensive assessment of the similarity between the images [25]. The SSIM value is between -1 and 1, where:

$$SSIM(I, \hat{I}) = \frac{\left(2\mu_I\mu_{\hat{I}} + C_1\right)\left(2\sigma_{I\hat{I}} + C_2\right)}{\left(\mu_I^2 + \mu_{\hat{I}}^2 + C_1\right)\left(\sigma_I^2 + \sigma_{\hat{I}}^2 + C_2\right)} \tag{13}$$

- RMSE (root mean square error): RMSE is a metric that evaluates the average error between predicted and actual values in a data set. It is expressed in the same unit as the target value. Here is the RMSE formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(I^i - \hat{I}^i\right)^2}{n}} \tag{14}$$

- MAE (mean absolute error): MAE is a metric used to measure the accuracy of a model by averaging the absolute errors between the values predicted by the model and the actual values. Unlike RMSE, MAE treats all errors equally without squaring them, making it a more intuitive measure for understanding the average error made by the model. The lower the MAE, the closer the model predictions are to the actual values. The MAE formula is as follows:

$$\bullet \quad MAE = \frac{1}{n}\sum_{i=1}^{n}\left|I^i - \hat{I}^i\right| \tag{15}$$

I: Real picture for sample.
$\hat{I}$: The predicted image for the sample.

4. Experiments and Results

For our experiment, we present the different spectral responses provided by the implemented models obtained from the 8-band MSFA one-shot cameras. We compare our models using the results obtained from our other metrics. The presentation of our results is conducted quantitatively with metrics and qualitatively.

4.1. Experimental Setting

The configuration of the algorithms used for setting up the models is detailed in the Table 1:

Table 1. Summary of algorithm configuration.

Setting	Values
Optimizer	Adam
Learning rate	10^{-3}
Loss function	Mean Squared error
Epochs	100
Batch size	8

4.2. Quantitative Results

Table 2 presents three models suitable for estimating the spectral response of 8-band MSFA one-shot cameras: VDSR, DRCN, and LapSRN, using several performance metrics.

Table 2. Summary of metric values for each model.

Models	Loss	PSNR	SSIM	SAM	RMSE	MAE
VDSR	0.0061	23.6255	0.8135	0.17293	0.06916	0.0525
DRCN	0.0047	25.5059	0.8355	0.13215	0.05849	0.0415
LapSRN	0.0057	23.9505	0.8047	0.17764	0.06534	0.04861

The loss function indicates the error between the predicted and actual images. The values obtained are 0.0061, 0.0047, and 0.0057 for the VDSR, DRCN, and LapSRN models. The DRCN model has the lowest loss, with 0.0047.

PSNR measures the quality of the reconstructed image, with higher values indicating better quality. DRCN achieves the highest PSNR at 25.5059, compared to VDSR at 23.6255 and LapSRN at 23.9505. SSIM evaluates the structural similarity between the original and reconstructed images. Values close to 1 indicate better similarity. DRCN also outperforms other models with an SSIM of 0.8355, while LapSRN has the lowest SSIM at 0.8047.

SAM measures the angle between spectral vectors, with lower values indicating better performance. DRCN has the lowest SAM value, 0.13215, indicating better spectral fidelity than VDSR, 0.17293, and LapSRN, 0.17764.

The RMSE quantifies the average magnitude of errors between the predicted and actual values. A lower RMSE value indicates better model accuracy. DRCN has the lowest RMSE, at 0.05849, while VDSR has the highest, at 0.06916.

MAE, which measures the average absolute errors in predictions, is a critical factor in evaluating the algorithms. DRCN's best performance, with an MAE of 0.0415, highlights its accuracy, while VDSR's highest MAE, at 0.0525, indicates its lower accuracy.

We can list hypotheses for which DRCN performs better:

DRCN (deeply recursive convolutional network) uses deeply recursive architectures that better capture complex and delicate details in images and could justify why DRCN outperforms other models in terms of PSNR, SSIM, and SAM, where accuracy and fidelity to detail are paramount.

In addition, its recursive mechanism would allow predictions to be refined at each iteration, thus capturing more nuances in image reconstruction. This capability could justify its better balance between noise reduction and texture preservation and its superior performance in terms of loss, RMSE, and MAE compared to VDSR and LapSRN models.

Figures 12–14 present the learning curves for the loss, PSNR, and SSIM metrics:

Figure 12 illustrates the learning and validation curves of the VDSR-based model, focusing on three key metrics: PSNR, SSIM, and the loss function (loss). These curves allow one to follow the evolution of the model's performance at each iteration during training.

Figure 13 illustrates the learning and validation curves of the DRCN-based model, focusing on three key metrics: PSNR, SSIM, and the loss function (loss). These curves allow one to follow the evolution of the model's performance at each iteration during training.

Figure 14 illustrates the learning and validation curves of the LapSRN-based model, focusing on three key metrics: PSNR, SSIM, and the loss function (loss). These curves allow one to follow the evolution of the model's performance at each iteration during training.

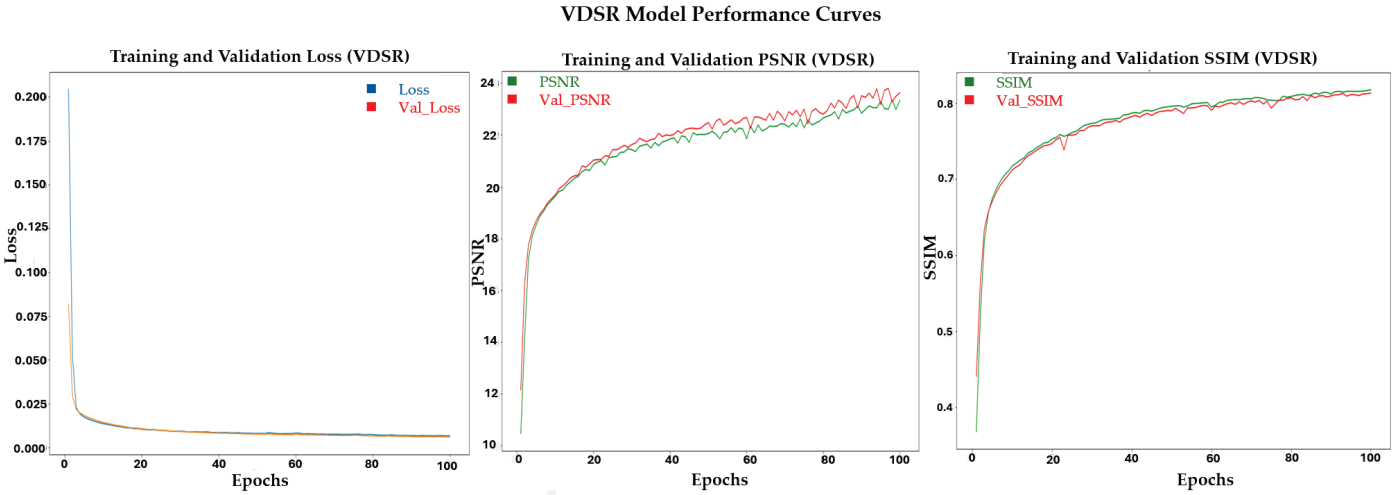


Figure 12. Presentation of the learning and validation curves of the VDSR model for the loss, PSNR, and SSIM metrics.

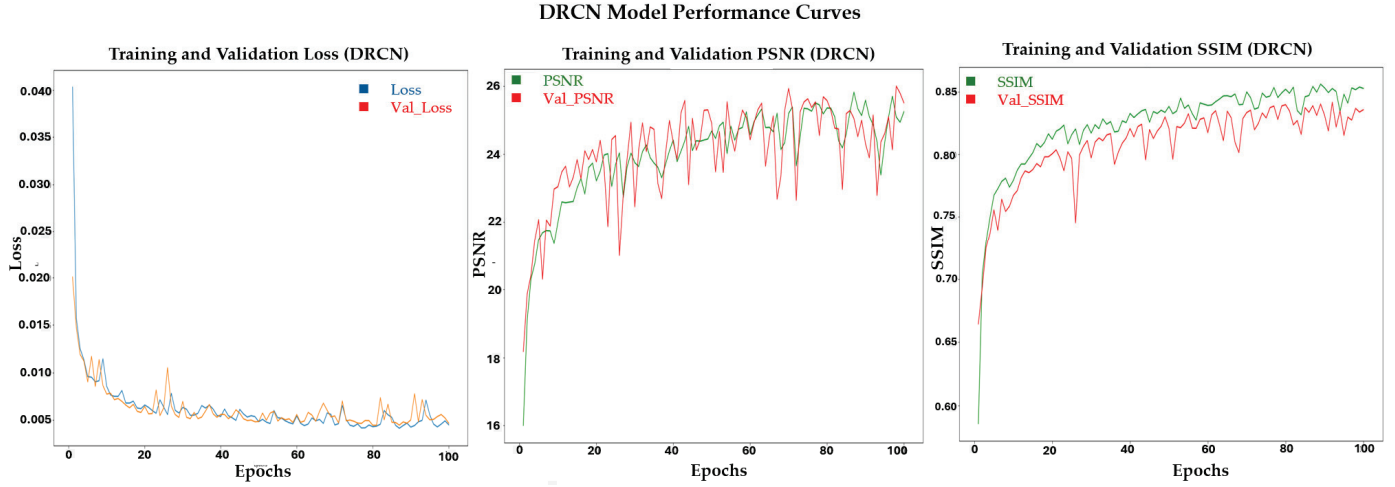


Figure 13. Presentation of the learning and validation curves of the DRCN model for the loss, PSNR, and SSIM metrics.

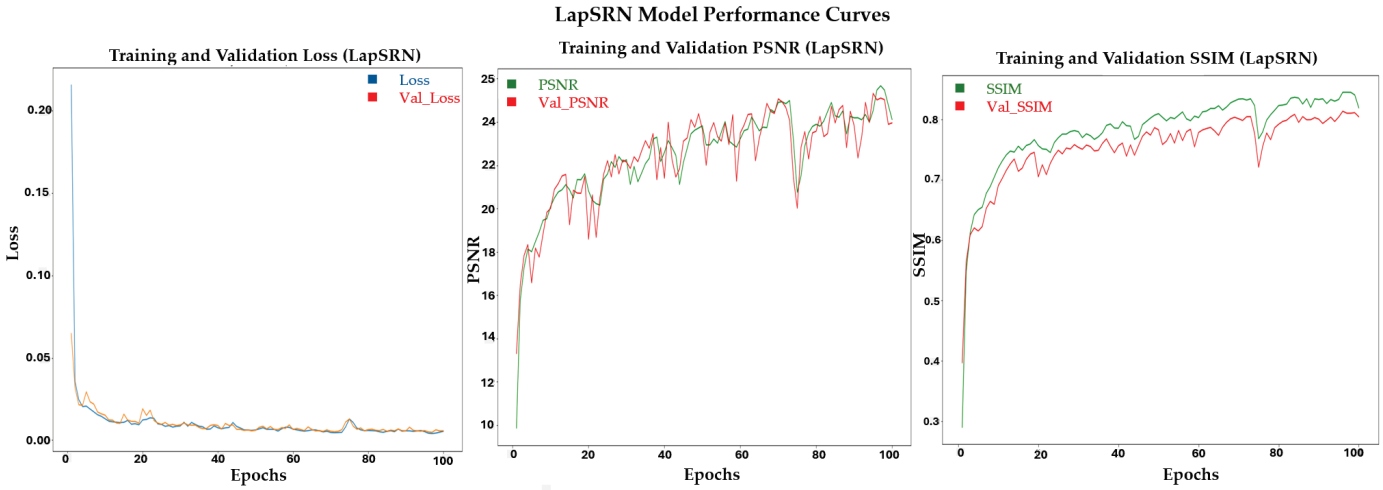


Figure 14. Presentation of the learning and validation curves of the LapSRN model for the loss, PSNR, and SSIM metrics.

4.3. Qualitative Results

This section presents the reference images (ground truth) and the spectral responses from the different models. For clarity, we focus on four specific wavelengths (420, 500, 670, 720) instead of the 31 bands available. This presentation focuses on three categories of images from the TokyoTech database: Chartres, Character, and Tshirts2, selected randomly.

Figures 15–17 show the spectral responses generated by the models and the ground truths.

We observed that the fitted DCRN model provides more accurate spectral responses than the fitted VDSR and LapSRN models, thus confirming previous quantitative results.

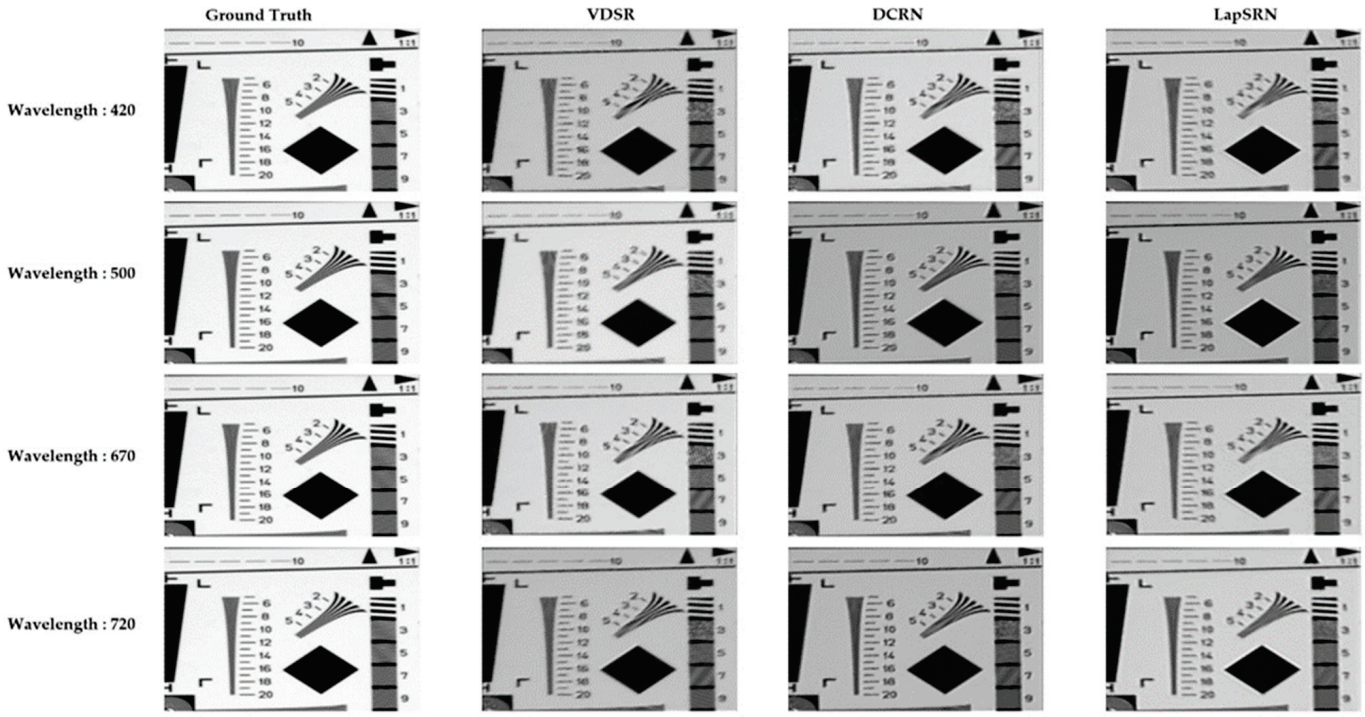


Figure 15. Presentation of the Chartres image, from the ground truth to the spectral response of the adapted VDSR, DCRN, and LapSRN models.

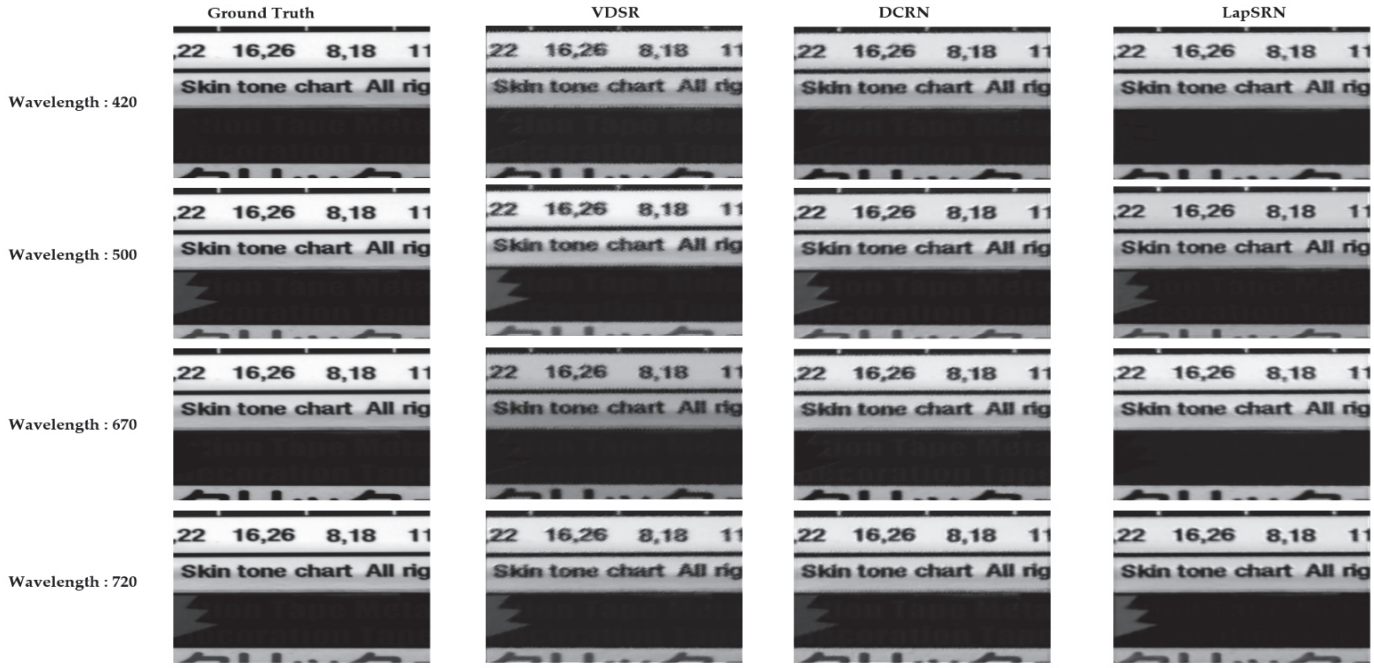


Figure 16. Presentation of the Character image, from the ground truth to the spectral response of the adapted VDSR, DCRN, and LapSRN models.

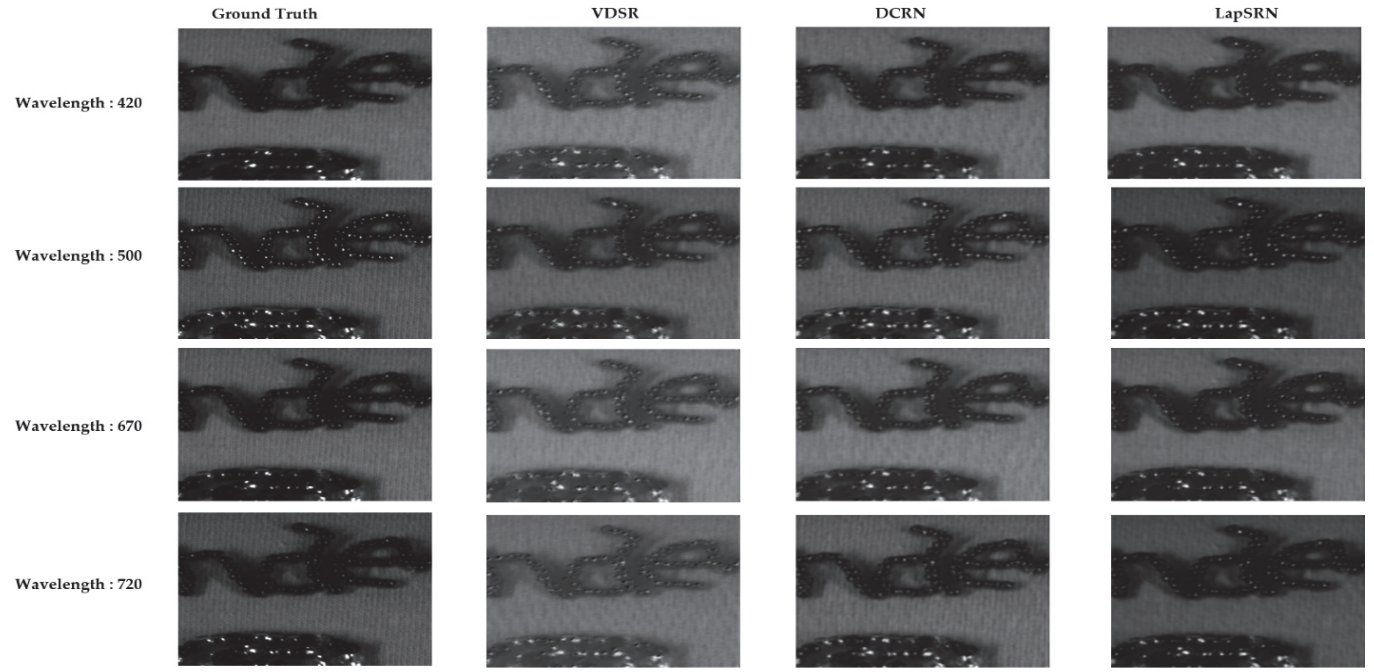


Figure 17. Presentation of the Tshirts2 image, from the ground truth to the spectral response of the adapted VDSR, DCRN, and LapSRN models.

5. Discussion

The performance of the adapted VDSR, DRCN, and LapSRN models reveals significant insights into their capabilities to estimate the spectral responses of 8-band one-shot MSFA cameras. Each model has distinct characteristics that directly influence their effectiveness, and the metric results provide valuable insight into their overall performance.

The DRCN model is the best of the three in quantitative and qualitative assessment. With the lowest values for loss, RMSE (root mean square error), and MAE (mean absolute error), as well as the highest values for PSNR (peak signal-to-noise ratio) and the SSIM

(structural similarity index), DRCN demonstrates an exceptional ability to minimize the error of spectral response estimation. These results show that DRCN does not just reduce the numerical differences between predictions and actual values and generate images perceived to be of better quality than the original images. DRCN's complex architecture enables deep learning of image features, which is the key to superior performance. The depth of the network, combined with advanced optimization techniques, allows DRCN to capture finer details and improve spectral fidelity, which is crucial for applications where color and texture accuracy is paramount.

Although not as efficient as DRCN, the LapSRN model shows interesting results in terms of error minimization (loss, RMSE, MAE) and produces slightly higher-quality images in terms of PSNR compared to VDSR. LapSRN is designed to minimize visual artifacts while maintaining reasonable accuracy, making it suitable for applications where visual quality is more important than structural or spectral fidelity.

VDSR, for its part, stands out for its ability to preserve structures and maintain high spectral precision, as shown by SSIM and SAM (spectral angle mapper) scores. Although its error minimization performance is not as impressive as that of DRCN or LapSRN, VDSR excels in faithfully reproducing details and fine features of images, which is crucial in scenarios where structural fidelity has priority.

Overall, DRCN emerges as the best-performing model for spectral response estimation among those evaluated, providing superior image quality and better spectral fidelity. However, selecting the appropriate model should not be based on overall performance alone; it must also consider the application's specific requirements, including available resources, use conditions, and priorities regarding visual quality, structural fidelity, or spectral precision. DRCN is ideal for applications requiring high accuracy and optimal image quality. At the same time, VDSR and LapSRN could be favored in contexts where other aspects, such as speed or simplicity of the model, are more important.

We use data covering wavelengths from 420 nm to 720 nm with an interval of 10 nm. This setting presents a first limitation: it becomes impossible to reproduce spectral variations lower than this 10 nm interval, which could restrict the precision of spectral reconstructions for applications requiring finer resolution. Furthermore, our results are limited to the visible spectrum since the available data only cover this range. Therefore, extending our findings to other spectral domains, such as infrared, would be infeasible with current data. Future exploration in these areas would require acquiring additional data in the corresponding wavelength ranges. These aspects constitute essential limitations for the interpretation of the results of this study.

Furthermore, the quality of the input data directly influences the results obtained. If the data used are of lower quality, with noise or artifacts, it becomes more difficult for models to accurately capture spectral and structural details. On the other hand, using very high-quality data would significantly improve model performance, leading to more accurate predictions and better spectral fidelity. It is, therefore, crucial to recognize that the size and quality of datasets play a key role in assessing model capabilities.

6. Threats to Validity

It is essential to consider some threats to the validity of this study, which could influence the interpretation of the results.

- The performance of the VDSR, DRCN, and LapSRN models strongly depends on the quality and representativeness of the data used. If the multispectral datasets used for training and validation are not sufficiently diverse or representative of real-world conditions, models could be overfit or underfit, limiting their ability to generalize to other types of cameras or multispectral scenes.
- Model performance also depends on the choice of hyperparameters, such as the networks' learning rates or depths. If these hyperparameters are not rigorously optimized for each model, comparisons may not reflect the maximum capabilities of each architecture.

- The conclusions drawn from this study are based on using a specific multispectral camera (8-band one-shot MSFA). It is important to recognize that these results may not be directly generalizable to other camera configurations, cameras with more or fewer spectral bands, or different application environments. For instance, environments that require more spectral precision or finer spatial resolution may yield different conclusions.

7. Conclusions

Spectral response evaluation of 8-band one-shot MSFA (multispectral filter array) cameras is an innovative research area combining multispectral imaging and advanced deep-learning techniques. In this study, we developed models based on VDSR, DRCN, and LapSRN algorithms, adapted to approximate the spectral response of 8-band one-shot MSFA cameras. We observed significant differences in performance and assessed using key metrics such as loss, PSNR, SSIM, SAM, RMSE, and MAE. The DRCN model performed best, achieving superior results on all metrics considered. Its ability to reduce reconstruction error while preserving the structural details of images makes it a preferred choice for applications requiring high image quality.

On the other hand, the VDSR and LapSRN models, although efficient, show lower performance, which could limit their use in contexts requiring optimal quality. Our study allowed us to develop models to estimate the spectral response of 8-band one-shot MSFA cameras into 31 bands. It is essential to continue this work to improve the quality of the spectral response because the quality of the original images significantly influences the results obtained. The future of image processing lies in developing models that combine performance, efficiency, and adaptability to various situations, ensuring continuous improvement in image quality in different application areas. Future research should focus on improving the input data, optimizing the models used, and exploring other models to find a better balance between performance and efficiency.

For future perspectives, the next step is to validate our approach to the camera developed as part of the CAVIAR project. For this purpose, a database is currently being prepared.

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

Funding: This research received no external funding.

Data Availability Statement: http://www.ok.sc.e.titech.ac.jp/res/MSI/MSIdata31.html (accessed date 14 October 2024).

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

References

1. Lanaras, C.; Baltsavias, E.; Schindler, K. Hyperspectral Super-Resolution with Spectral Unmixing Constraints. *Remote Sens.* **2017**, *9*, 1196. [CrossRef]
2. Stuart, M.B.; Davies, M.; Hobbs, M.J.; Pering, T.D.; McGonigle, A.J.S.; Willmott, J.R. High-Resolution Hyperspectral Imaging Using Low-Cost Components: Application within Environmental Monitoring Scenarios. *Sensors* **2022**, *22*, 4652. [CrossRef] [PubMed]
3. Cao, S.-Y.; Shen, H.-L.; Chen, S.-J.; Li, C. Boosting Structure Consistency for Multispectral and Multimodal Image Registration. *IEEE Trans. Image Process.* **2020**, *29*, 5147–5162. [CrossRef]
4. Mohammadi, V.; Gouton, P.; Rossé, M.; Katakpe, K.K. Design and Development of Large-Band Dual-MSFA Sensor Camera for Precision Agriculture. *Sensors* **2024**, *24*, 64. [CrossRef] [PubMed]
5. Diaz, N.; Alvarado, A.; Meza, P.; Guzmán, F.; Vera, E. Multispectral Filter Array Design by Optimal Sphere Packing. *IEEE Trans. Image Process.* **2023**, *32*, 3634–3649. [CrossRef] [PubMed]

6. Pu, R. Mapping Tree Species Using Advanced Remote Sensing Technologies: A State-of-the-Art Review and Perspective. *J. Remote Sens.* **2021**, *2021*, 2021–9812624. [CrossRef]
7. Monno, Y.; Kikuchi, S.; Tanaka, M.; Okutomi, M. A Practical One-Shot Multispectral Imaging System Using a Single Image Sensor. *Trans. Image Proc.* **2015**, *24*, 3048–3059. [CrossRef] [PubMed]
8. Toivonen, M.E.; Klami, A. Practical Camera Sensor Spectral Response and Uncertainty Estimation. *J. Imaging* **2020**, *6*, 79. [CrossRef] [PubMed]
9. Darrodi, M.M.; Finlayson, G.; Goodman, T.; Mackiewicz, M. Reference Data Set for Camera Spectral Sensitivity Estimation. *J. Opt. Soc. Am. A* **2015**, *32*, 381. [CrossRef] [PubMed]
10. Han, S.; Matsushita, Y.; Sato, I.; Okabe, T.; Sato, Y. Camera Spectral Sensitivity Estimation from a Single Image under Unknown Illumination by Using Fluorescence. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 805–812.
11. Zhao, H.; Rei, K.; Tan, R.; Ikeuchi, K. Estimating Basis Functions for Spectral Sensitivity of Digital Cameras. In Proceedings of the Image Recognition and Understanding, Kyoto, Japan, 20–22 July 2009.
12. Prasad, D.; Nguyen, R.; Brown, M. Quick Approximation of Camera's Spectral Response from Casual Lighting. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Sydney, NSW, Australia, 2–8 December 2013; pp. 844–851.
13. Matanga, J.; Gouton, P.; Bourillot, E.; Journaux, S.; Ele, P. Method for the Optimal Approximation of the Spectral Response of Multicomponent Image. *Electron. Imaging* **2019**, *31*, 1–9. [CrossRef]
14. Tian, C.; Zhang, X.; Lin, J.C.-W.; Zuo, W.; Zhang, Y.; Lin, C.-W. Generative Adversarial Networks for Image Super-Resolution: A Survey. *arXiv* **2022**, arXiv:2204.13620.
15. Vizilter, Y.V.; Vygolov, O.V.; Zheltov, S.Y.; Rubis, A.Y. A Way to Interconnect the Multispectral Images for Improved Vision Systems by Using the Methods of Diffusive Morphology. *J. Comput. Syst. Sci. Int.* **2016**, *55*, 598–608. [CrossRef]
16. Andriyanov, N.A.; Vasiliev, K.K.; Dement'ev, V.E. Analysis of the Efficiency of Satellite Image Sequences Filtering. *J. Phys. Conf. Ser.* **2018**, *1096*, 012036. [CrossRef]
17. Andriyanov, N.A.; Vasiliev, K.K.; Dementiev, V.E.; Belyanchikov, A.V. Restoration of Spatially Inhomogeneous Images Based on a Doubly Stochastic Model. *Optoelectron. Instrument. Proc.* **2022**, *58*, 465–471. [CrossRef]
18. Yao, J.Y.A.; Ayikpa, K.J.; Gouton, P.; Kone, T. A Multi-Shot Approach for Spatial Resolution Improvement of Multispectral Images from an MSFA Sensor. *J. Imaging* **2024**, *10*, 140. [CrossRef] [PubMed]
19. Christopher, A.; Hari Kishan, R.; Sudeep, P.V. Image Reconstruction Using Deep Learning. In *Machine Learning Algorithms for Signal and Image Processing*; IEEE: New York, NY, USA, 2023; pp. 65–87. ISBN 978-1-119-86183-6.
20. Kim, J.; Lee, J.K.; Lee, K.M. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1646–1654.
21. Kim, J.; Lee, J.K.; Lee, K.M. Deeply-Recursive Convolutional Network for Image Super-Resolution. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1637–1645.
22. Zhang, X.; Song, H.; Zhang, K.; Qiao, J.; Liu, Q. Single Image Super-Resolution with Enhanced Laplacian Pyramid Network via Conditional Generative Adversarial Learning. *Neurocomputing* **2020**, *398*, 531–538. [CrossRef]
23. Lai, W.-S.; Huang, J.-B.; Ahuja, N.; Yang, M.-H. Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5835–5843.
24. Spectral Angle Mapper. Available online: https://www.nv5geospatialsoftware.com/docs/SpectralAngleMapper.html (accessed on 13 August 2024).
25. Wang, Z.; Bovik, A.; Sheikh, H. *Structural Similarity Based Image Quality Assessment. Digital Video Image Quality and Perceptual Coding*; Series in Signal Processing and Communications; CRC Press: Boca Raton, FL, USA, 2005. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Improving Re-Identification by Estimating and Utilizing Diverse Uncertainty Types for Embeddings

Markus Eisenbach *, Andreas Gebhardt, Dustin Aganian and Horst-Michael Gross

Neuroinformatics and Cognitive Robotics Lab, Ilmenau University of Technology, 98693 Ilmenau, Germany; dustin.aganian@tu-ilmenau.de (D.A.); horst-michael.gross@tu-ilmenau.de (H.-M.G.)
* Correspondence: markus.eisenbach@tu-ilmenau.de

Abstract: In most re-identification approaches, embedding vectors are compared to identify the best match for a given query. However, this comparison does not take into account whether the encoded information in the embedding vectors was extracted reliably from the input images. We propose the first attempt that illustrates how all three types of uncertainty, namely model uncertainty (also known as epistemic uncertainty), data uncertainty (also known as aleatoric uncertainty), and distributional uncertainty, can be estimated for embedding vectors. We provide evidence that we do indeed estimate these types of uncertainty, and that each type has its own value for improving re-identification performance. In particular, while the few state-of-the-art approaches that employ uncertainty for re-identification during inference utilize only data uncertainty to improve single-shot re-identification performance, we demonstrate that the estimated model uncertainty vector can be utilized to modify the feature vector. We explore the best method for utilizing the estimated model uncertainty based on the Market-1501 dataset and demonstrate that we are able to further enhance the performance above the already strong baseline UAL. Additionally, we show that the estimated distributional uncertainty resembles the degree to which the current sample is out-of-distribution. To illustrate this, we divide the distractor set of the Market-1501 dataset into four classes, each representing a different degree of out-of-distribution. By computing a score based on the estimated distributional uncertainty vector, we are able to correctly order the four distractor classes and to differentiate them from an in-distribution set to a significant extent.

Keywords: uncertainty estimation for embedding vectors; model uncertainty; data uncertainty; distributional uncertainty; re-identification; out-of-distribution

1. Introduction

The development of self-driving cars [1,2] and robots for collaborative assembly [3,4] represents a significant advancement in autonomous systems. However, these technologies rely on intelligent components to perceive their environment, especially for perception of humans and interaction with them. In the current state of the art, the majority of perception systems are based on neural networks, which, like many machine learning methods, compute outputs for any input, regardless of whether they can make safe decisions or not. Therefore, estimating the uncertainty in the decision-making process is crucial for the development of a reliable autonomous system that can assist humans without causing harm. As a result, uncertainty quantification or estimation for neural networks has emerged as a growing research trend in recent years.

Starting from basic work on uncertainty quantification [5,6], work on uncertainty estimation for neural networks has been mainly focused on classification and regression tasks [7–10]. This includes perception tasks such as object detection [11–14], semantic and medical image segmentation [15–18], as well as action recognition [19], in which actions and objects are classified, and localization is achieved through bounding box regression. The determined uncertainty in these cases relates to the final classification or regression layer.

However, there is currently a trend to use neural networks as encoders to describe images or language as feature vectors, which is known as learning a feature embedding. The estimation of uncertainties for embedding vectors is much less explored [8].

To quantify uncertainties for feature vectors, most state-of-the-art approaches use probabilistic embeddings [20,21], which are able to estimate data uncertainty. Only few approaches [22–25] estimate model uncertainty for feature vectors by using a Bayesian module on top of the deterministic backbone. These are the two most relevant types of uncertainty in literature [26,27]. However, for autonomous agents it is also important to recognize when the data are out-of-distribution, which means that the current input is very dissimilar from data seen during training. Then, it is likely that the agent cannot make safe decisions even if model uncertainty and data uncertainty are low. If such situations are relevant, according to literature, uncertainty can be categorized into three types of uncertainty [7]: (1) model uncertainty (also known as epistemic uncertainty), (2) data uncertainty (also known as aleatoric uncertainty), and (3) distributional uncertainty. Neither probabilistic embeddings nor Bayesian modules are able to quantify distributional uncertainty. Indeed, to the best of our knowledge, the estimation of distributional uncertainty for feature vectors has not been explored at all.

We show that it is beneficial to estimate all three types of uncertainty and how each of the uncertainty types has its individual benefits for predicting different types of error sources for the feature vector. Notably, this information can be used to improve the embedding during training and inference.

In particular, we demonstrate how a component-wise *model uncertainty* estimate can be leveraged to enhance the feature vector during inference. Furthermore, we show that during training estimated *data uncertainty* can be employed to scale the embedding vector elements in accordance with the degree of uncertainty, as proposed by Dou et al. [25]. Additionally, we illustrate how estimated *distributional uncertainty* can be utilized to determine the extent to which an input is out-of-distribution.

While our approach can be used in general to estimate and utilize the three types of uncertainty, for the purposes of benchmarking, we decided in favor of person re-identification as the application. Person re-identification represents an exemplary human perception task that is relevant for autonomous agents that require the ability to track individuals or interact with humans. The selection of person re-identification for benchmarking is consistent with related work on uncertainty estimation for feature vectors [22,23,28], which made the same decision since person re-identification provides well-defined benchmarks while the task is very challenging. Furthermore, it is one of the few domains, where uncertainties for embedding vectors have been employed to enhance decision-making, thereby enabling a comparison of our results with multiple state-of-the-art approaches.

While the state-of-the-art approaches that make use of uncertainties for person re-identification during inference only use data uncertainty to improve the single-shot person re-identification performance, in our experiments, we show that we can use the other two types of uncertainty to further improve the performance over the strong baseline and to quantify the extent to which an input image is out-of-distribution.

Our contributions are as follows:

1. In contrast to related work, we estimate three different types of uncertainty for feature vectors and, based on our experiments, we emphasize the individual value of diverse uncertainty types for improving the embedding and thereby the re-identification.
2. We show that our *distributional uncertainty* estimate enables us to measure the degree to which an input is out-of-distribution, which, based on literature, is a strong indicator that the extracted values actually represent distributional uncertainty.
3. We utilize the estimated *model uncertainty* vector to perform element-wise modifications to the feature vector, and thereby improve the person re-identification performance notably over an already strong baseline.

This paper is organized as follows: In Section 2, we describe the state of the art on uncertainty quantification for feature vectors and the utilization of this uncertainty

information. In Section 3, we describe the mathematics of uncertainty quantification and explain the algorithm for estimating three different types of uncertainty. In Section 4, we evaluate our experiments regarding the usefulness of the estimation and utilization of diverse types of uncertainty. Finally, in Section 5, we draw a conclusion and provide an outlook on future work.

2. Related Work

The majority of related work on uncertainty quantification focuses on classification and regression tasks [7–10]. In contrast, the estimation of uncertainties for embedding vectors is much less explored [8].

In the following, we describe, how uncertainty is quantified for feature vectors (Section 2.1) and how this uncertainty information can be utilized to improve the feature embedding (Section 2.2).

2.1. Uncertainty Quantification for Feature Vectors

In literature, mainly the quantification of data uncertainty (also known as aleatoric or statistical uncertainty) and model uncertainty (also known as epistemic or systemic uncertainty) is considered. Additionally, the quantification of distributional uncertainty has been proposed by Malinin and Gales [26] and Nandy et al. [27].

2.1.1. Quantification of Data Uncertainty

To estimate uncertainties for feature vectors, in related work, most often probabilistic embeddings [21] are used, which are sometimes also referred to as stochastic embeddings [29].

Probabilistic Embeddings for Data Uncertainty Estimation

Probabilistic embeddings were first introduced for natural language processing [20,30] and later, based on the example of face recognition [21,31], also for computer vision tasks. The key idea of probabilistic embeddings is to estimate a mean and a variance vector instead of a point estimate for the feature vector. The variance vector is interpreted as a diagonal covariance matrix. Therefore, each output is represented as a multivariate normal distribution in the embedding space. The mean vector represents the feature vector and the variance vector represents the data uncertainty of the feature vector elements. Probabilistic embeddings have been widely used for computer vision tasks, e.g., for face recognition [21,31–33], few-shot recognition [29], emotion recognition [34], human pose embeddings [35], image retrieval [36–38], 3D shape retrieval [39], point cloud instance segmentation [40], self-supervised and representation learning [41–46], multi-source domain adaptation [47], vision-language embedding alignment [48–50], and person re-identification [25,28,51–54]. In the majority of these approaches, a Gaussian distribution has been used to model the uncertainty. Only few approaches [28,33] present methods for deviating from a Gaussian distribution.

To get a definite-valued output for further processing, most of these works sample from this distribution, which is also the way variational autoencoders (VAEs) [55] handle the distribution in the latent space. Some approaches [21,52,56] use the distributions directly. Dou et al. [25] use the variance information to weight the mean vector.

The estimated uncertainty in probabilistic embeddings represents only data uncertainty. Therefore, these probabilistic embeddings cannot quantify model uncertainty or distributional uncertainty by themselves.

Alternative to Probabilistic Embeddings

To date, only one alternative to probabilistic embeddings has been proposed for the estimation of data uncertainty. Taha et al. [23] propose a method for estimating a kind of data uncertainty, which they then use to compute uncertainty-based weighting factors in the triplet loss. However, the uncertainty they estimate does not correspond to the

mathematical model of data uncertainty. Therefore, probabilistic embeddings represent the only mathematically grounded approach for estimating data uncertainty for feature vectors that has been proposed in the state of the art.

2.1.2. Quantification of Model Uncertainty

Quantifying model uncertainty for feature vectors is addressed rarely. Approaches have only been proposed for image captioning [24] and person re-identification [22,23,25].

Bayesian Neural Networks for Model Uncertainty Estimation

All these approaches [22–25] use Bayesian Neural Networks (BNNs) to estimate the model uncertainty. BNNs replace scalar weights with a distribution over them and then use approaches such as Markov Chain Monte Carlo or Variational Inference to update the distribution such that a good set of weights becomes more and more likely. They are not very popular in deep neural networks, since they suffer from performance problems and are hard to train.

As a compromise to approximate model uncertainty with limited extra computation, applying a Bayesian component on top of a deterministic backbone is a popular choice. Indeed, the re-identification approaches [22,23,25] all use a Bayesian module on top of a deterministic backbone. In this Bayesian module, Monte Carlo Dropout is used to sample K models to estimate the model uncertainty.

Joint Estimation of Data and Model Uncertainty

Only few approaches [23,25] attempt to jointly estimate both data uncertainty and model uncertainty for feature vectors. We use the Uncertainty Aware Learning (UAL) approach [25] as our baseline since it uses a reasonable choice of a Bayesian module to estimate model uncertainty and a mathematically grounded approach to estimate data uncertainty by using a probabilistic embedding.

2.1.3. Quantification of Distributional Uncertainty

To the best of our knowledge, there is no related work that has estimated distributional uncertainty for feature vectors, so far. Basic studies, like the ones of Malinin and Gales [26] and Nandy et al. [27], propose methods for estimating distributional uncertainty for a classification task.

2.2. Utilization of Uncertainties

Estimated uncertainties can be utilized during training to improve the learning process and during inference to filter out unsuitable inputs, to fuse multi-model or multi-shot data, as well as to improve the feature vector.

2.2.1. Utilization of Uncertainty during Training

One thing in common with most works that attempt to estimate uncertainty in some way is that they let their uncertainty estimates flow into the loss function in order to improve learning, hoping to enable the model to learn a better embedding.

In the event that some training data are unsuitable and hinder optimal training progress, it is beneficial to restrict their influence. This can be achieved by incorporating an estimate of data uncertainty. In accordance with this principle, the estimated data uncertainty from a probabilistic embedding is often employed to implicitly enhance the embedded features during training, thereby facilitating an improvement in feature matching [21,25,28,31–33,36–38,51,53,54].

The estimation of model uncertainty with a Bayesian module has also been shown to improve the embedding during training [22,23,25].

2.2.2. Utilization of Uncertainty during Inference

During inference, estimated uncertainty for feature vectors can be used to filter out unsuitable inputs, to fuse multi-model or multi-shot data, and to improve the feature vector.

Utilization of Uncertainty to Filter Out Unsuitable Inputs

Many approaches restrict their utilization of uncertainty to the calculation of a scalar uncertainty quantifier, also known as a quality or reliability score. This scalar uncertainty quantifier is subsequently employed to identify low-quality or out-of-distribution inputs [21,25,26,28,57] in order to filter them out.

One common use case is the visualization of the estimated uncertainty for monitoring purposes. However, the awareness of unsuitable inputs is also a beneficial piece of information for an autonomous agent to recognize that the input is likely different from the training data. This may occur when an object is cropped based on an imprecise bounding box. In such a case, the autonomous agent may attempt to obtain a more accurate representation by capturing the scene from a different perspective, rather than relying on the current input to make a decision. Similarly, in the context of person re-identification, if an autonomous agent tracks its user based on a feature vector and other cues, as in the framework of Müller et al. [58], the uncertainty estimate is a valuable indicator to determine when a feature vector should be excluded from the tracking process, as it may be out-of-distribution.

Utilization of Distributional Uncertainty to Detect Out-of-Distribution Data

For a classification task, Malinin et al. [26] show the value of distributional uncertainty to recognize cases, where the input is far away from the distribution of examples in the training data. This field of out-of-distribution detection has been the subject of much research in recent years [59,60], but with the sole focus on classification tasks.

Utilization of Uncertainty for Feature Vectors to Detect Out-of-Distribution Data

Schwaiger et al. [57] examined whether estimated model uncertainty or data uncertainty for feature vectors can be used to identify out-of-distribution samples. These types of estimated uncertainty were not good enough to reliably detect out-of-distribution data. In our experiments, we confirm that these two types of estimated uncertainty are only somewhat suitable for out-of-distribution detection and that estimated distributional uncertainty is a much better fit.

Utilization of Uncertainty for Multi-Modality, Multi-View, and Multi-Shot

If the goal is to compute a joint embedding for vision and language inputs [48–50], estimated data uncertainty helps to better align the embeddings from different modalities.

To weight different samples for the same instance, data uncertainty and model uncertainty have been used to fuse multiple shots in re-identification [21,25,61], multiple views [62], or multiple modalities [63]. Please note that multi-shot approaches are not part of our examinations. In this paper, we discuss the value of estimated uncertainty for single-shot re-identification.

Utilization of Uncertainty to Improve the Feature Vector

In order to compute the feature vector, Dou et al. [25] divide the estimated mean tensor of the probabilistic embedding by the estimated variance tensor of the probabilistic embedding. Therefore, they include data uncertainty in the computation of the feature vector during inference, which represents an improvement over the baseline.

Other than for multi-shot purposes and filtering of unsuitable data, estimated model uncertainty for feature vectors has not been utilized during inference so far.

3. Uncertainty Estimation in Neural Networks — Mathematics and Algorithm

In deep-learning-based re-identification (ReID), we use a deep neural network (DNN) to map some input $x \in \mathcal{X}$ to an abstract representation $z \in \mathcal{Z}$, known as the embedding, that we can efficiently compare to other embeddings in order to find inputs that are similar to each other. The DNN is thus a function $f_\theta : \mathcal{X} \to \mathcal{Z}$, parameterized by its weights θ. Generally, $\mathcal{Z} = \mathbb{R}^d$ and z is therefore called the d-dimensional feature vector or embedding vector.

In Section 3.1, we introduce the mathematical theory underlying the estimation of uncertainty for feature vectors. In Section 3.2, we present the motivation for extracting different types of uncertainty. In Section 3.3, we propose an architecture for the extraction of three types of uncertainty. In Section 3.4, we discuss the alignment between the estimated uncertainty in our architecture and the mathematical theory.

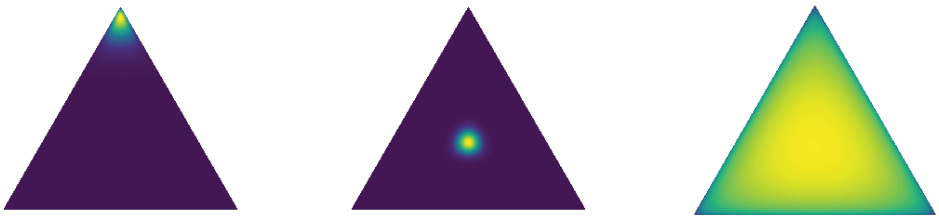
3.1. Mathematical Theory on Modeling Uncertainty

As detailed in the analysis conducted by Gawlikowski et al. [7], there are numerous points throughout the preparation, design, and training of a DNN where errors may occur and uncertainty arises. This uncertainty is subsequently reflected in the predictive uncertainty, which is modeled as a probability distribution $p(z)$ over the model outputs, specifically the embedding vector in this case. In order to gain further insight into this somewhat vague concept, it is helpful to decompose this general uncertainty into sub-types, based on its source.

3.1.1. Distinction of Model, Data, and Distributional Uncertainty

Model uncertainty is inherent to the model and is caused by problems with the model structure or training process, such as insufficient model complexity, training duration, or dataset composition. Data uncertainty is inherent to the data and is caused by information loss in the measuring process, such as limited image resolution, motion blur, or labeling errors. Distributional uncertainty is intended to cover the concept of out-of-distribution (OOD) inputs.

Even when a model is perfectly tuned for its task and the data is flawless (low model and data uncertainty), the predictions may still be unreliable if the model is applied to inputs it was not designed to handle (high distributional uncertainty). For example, if a person ReID model is provided with images of dogs, bikes, or text, unexpected results may be produced. Figure 1 illustrates the role of *distributional uncertainty* in contrast to the other uncertainty types.



(**a**) Confident Prediction (**b**) High Data Uncertainty (**c**) High Distributional Uncertainty

Figure 1. Example for a Dirichlet distribution with three classes in three scenarios. Points on the simplex correspond to a parameter configuration (p_1, p_2, p_3) of the categorical distribution $(p(i) = p_i)$. Yellow (blue) corresponds to a high (low) probability for the configuration. (**a**) The configuration $p_1 \approx 1,\ p_2, p_3 \approx 0$ is most likely. (**b**) The configuration $p_1 \approx p_2 \approx p_3$ is most likely. (**c**) All configurations are equally likely. Image source: Malinin and Gales [26], similar visualization used by Gawlikowski et al. [7].

In the case of high data uncertainty and low distributional uncertainty (Figure 1b), we know that we do not know which class is presented, whereas in the case of high distributional uncertainty (Figure 1c), we do not know whether we know and thus should not trust any seemingly certain prediction of category or level of data uncertainty we might

get. One could say that distributional uncertainty arises due to the interaction between data and model, specifically due to a mismatch between them. In addition to model uncertainty and data uncertainty, we will also consider distributional uncertainty in this work.

3.1.2. Mathematical Model

A universally accepted mathematical model for uncertainty in neural networks has yet to be established, and it is possible that one may never emerge. This is because, as Kiureghian et al. [64] argue, the specific model may be context-dependent. Malinin and Gales [26] adapt the more common model proposed by Gal and Ghahramani [65] to include distributional uncertainty, and arrive at Equation (1). Their approach is classification-based, and thus the predictive uncertainty is modeled as a distribution over the class prediction $\hat{y}$ instead of the embedding vector.

$$\underbrace{p(\hat{y} \mid x, \mathcal{D})}_{\text{Uncertainty}} = \int\int \underbrace{p(\hat{y} \mid \xi)}_{\substack{\text{Data} \\ \text{Uncertainty}}} \cdot \underbrace{p(\xi \mid x, \theta)}_{\substack{\text{Distributional} \\ \text{Uncertainty}}} \cdot \underbrace{p(\theta \mid \mathcal{D})}_{\substack{\text{Model} \\ \text{Uncertainty}}} \, \mathrm{d}\xi \mathrm{d}\theta \tag{1}$$

Here, x is the input, $\mathcal{D}$ is the training dataset, and θ represents the model parameters. The Dirichlet distribution $p(\xi \mid x, \theta)$ is a distribution over the categorical distribution $p(\hat{y} \mid \xi)$, where ξ acts as a stand-in for the distribution $p(\hat{y} \mid \xi)$ and can in practice be thought of as its parameters. This formulation is not ideally suited to embedding-based tasks, as it models uncertainty based on the class prediction $\hat{y}$. Consequently, we adapt Equation (1) to be based in the embedding space, resulting in Equation (2), where z is the embedding vector as defined above.

$$\underbrace{p(z \mid x, \mathcal{D})}_{\text{Uncertainty}} = \int\int \underbrace{p(z \mid \xi)}_{\substack{\text{Data} \\ \text{Uncertainty}}} \cdot \underbrace{p(\xi \mid x, \theta)}_{\substack{\text{Distributional} \\ \text{Uncertainty}}} \cdot \underbrace{p(\theta \mid \mathcal{D})}_{\substack{\text{Model} \\ \text{Uncertainty}}} \, \mathrm{d}\xi \mathrm{d}\theta \tag{2}$$

Now, data uncertainty is modeled as a distribution in the embedding space, distributional uncertainty as a distribution over those distributions (i.e., their parameters), and model uncertainty as a distribution over the model parameters. This is consistent with the modelling approach employed in previous works that utilize probabilistic embeddings, i.e., distributions in the embedding space, to capture data uncertainty [20,21,25,28–54] (see Section 2.1.1), as well as Bayesian Neural Networks, in which distributions are placed over the model parameters, to capture model uncertainty [22–25] (see Section 2.1.2).

Note that the color scheme for data, distributional, and model uncertainty, as well as the feature vector is used throughout this work to further indicate what is being displayed.

3.2. Value of Quantified Uncertainty

The mathematical theory offers insight into the distinction between three types of uncertainty. This subsection aims to provide readers with a comprehensive understanding of the individual value of these three types of uncertainty. Readers who are already familiar with this topic may choose to skip this subsection.

The most straightforward value of quantified uncertainty is the measure of the extent to which the output of the model can be trusted. Such information can be provided to decision-makers, users, subsequent processing modules in autonomous agents, or to the model itself, with the objective of improving the learned representation during training. The deployment of uncertainty as a metric for trustworthiness is of particular significance in scenarios where critical decisions must be made, as misjudgments can result in substantial consequences, including damage or injury. One such example is the interaction between autonomous agents and humans. When high uncertainty is observed, it indicates *that* the system's output should not be trusted. However, the reason *why*, such predictions

should not be trusted is not immediately apparent. To uncover the underlying reason, it is necessary to decompose the uncertainty into different types, which we will discuss below.

3.2.1. Individual Value of Model Uncertainty

The value of quantified model uncertainty lies in the acquisition of information regarding the competence of the model with respect to the task at hand. In particular, it indicates how likely it is that the model makes errors when operating in laboratory conditions.

If the model uncertainty indicates that the model is deficient in its ability to perform the task at hand, we may attempt to improve the model. This may be achieved, for instance, by providing additional training data for scenarios in which the model is uncertain. Moreover, model uncertainty may serve as a valuable indicator of the model's convergence state during training. If the model has not converged on some of the training data, an estimate of model uncertainty can be utilized to regulate the extent to which the model should learn from a particular set of training examples.

Model uncertainty is also the foundation for distributional uncertainty. If the model is not adequately tuned to its task, we should not rely on its capacity to accurately predict when a given situation aligns with this task. This is also evident in the mathematical model (Equation (2)) where the distributional uncertainty term is conditioned upon the model uncertainty term.

3.2.2. Individual Value of Distributional Uncertainty

The value of quantified distributional uncertainty lies in the acquisition of information regarding the expected competence of the model in a given situation. Therefore, distributional uncertainty measures the extent to which the model's expertise aligns with the problem it is attempting to solve in the current situation.

If distributional uncertainty indicates a limitation in the model's ability to address the current situation, the input can be delegated to an alternative model or a human operator. Alternatively, this information can be provided to any subsequent processing modules of an autonomous agent that depend on the accuracy of the model output.

Distributional uncertainty also serves as a foundation for estimating data uncertainty. If the model's expertise does not match the current situation, it is not reasonable to rely on it to assess the quality of the data. This is also reflected in the mathematical model (Equation (2)) where the data uncertainty term is conditioned upon the distributional uncertainty term.

3.2.3. Individual Value of Data Uncertainty

The value of quantified data uncertainty lies in the acquisition of information regarding the theoretical limits of prediction accuracy. Even if the model were perfectly constructed for its task and if the current situation perfectly matched that task, it might still be impossible to make certain predictions if the data on which they are based is flawed.

In such cases, data uncertainty indicates that the data does not contain sufficient information to make a precise prediction. In these instances, it is possible to gather better data or otherwise make a nuanced prediction, specifying multiple possibilities that could be true. Furthermore, if data uncertainty is modeled as a probabilistic embedding, the inherent uncertainty in the data, such as missing details due to resolution or motion blur, can be transferred into the embedding space.

In practice, the boundary between data uncertainty and distributional uncertainty can be quite fuzzy. An image of a person that has been wrongly cropped, displaying only an arm, could be perceived as exhibiting high data uncertainty due to the erroneous setting of the crop. However, it could also be regarded as exhibiting high distributional uncertainty, given that images of arms belong to a distinct domain from images of people. Similarly, the introduction of minor amounts of noise into an image would be regarded as data uncertainty. However, if the noise is so extensive that the original image is no longer recognizable, this would represent an instance of the domain of images of noise, which would be indicated by high distributional uncertainty.

In summary, the information obtained allows us to identify the sources of uncertainty. We can distinguish whether the data are ambiguous (data uncertainty), the model is unable to cope with the current situation (distributional uncertainty), or the model is underperforming even under ideal conditions (model uncertainty).

3.3. Uncertainty Estimation for Embedding Vectors

The basis for our architecture to estimate the three types of uncertainty is the Uncertainty Aware Learning (UAL) approach [25]. The structure of our architecture is given in Figure 2 (detail) and Figure 3 (overview).

As shown in Figure 3, the basic feature extraction is performed by the backbone f, which processes the input $x \in \mathbb{R}^{3\times256\times128}$ and extracts a feature tensor $F \in \mathbb{R}^{d\times16\times8}$ (Equation (3)). Therefore, as a result of the downsampling operations in the backbone, the height and width of the feature map are reduced to $\frac{1}{16}$ of the height and width of the input image (see Figure 3).

$$F = f(x) \tag{3}$$

The backbone is a ResNet-50 based on the BOT approach [66], which is pre-trained on ImageNet. In this case, the number of channels in the feature tensor F is $d = 2048$. The feature tensor F is then processed by dedicated modules—the Bayesian module and the embedding heads—to estimate the three types of uncertainty in addition to the feature vector.

The Bayesian module enables the estimation of *model uncertainty*. The embedding heads realize a probabilistic embedding, which enables the estimation of *data uncertainty*. The combination of the Bayesian module and the variance head enables the estimation of *distributional uncertainty*. All modules for uncertainty estimation are initialized randomly. Their structure is detailed in Figure 2, and details about their realization are presented below.

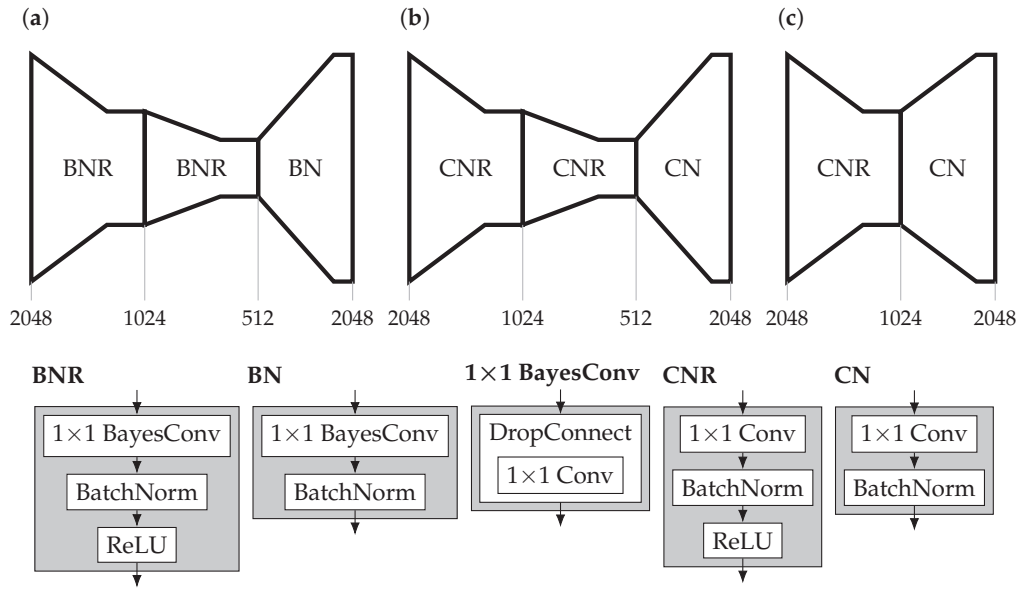


Figure 2. Structure of (**a**) the Bayesian module, (**b**) the mean head, and (**c**) the variance head (top row) and of their building blocks (bottom row). Underneath the diagrams in (**a**–**c**), the number of channels at each stage is indicated. Abbreviations: B—Bayes convolution, C—standard convolution, N—batch normalization, R—ReLU.

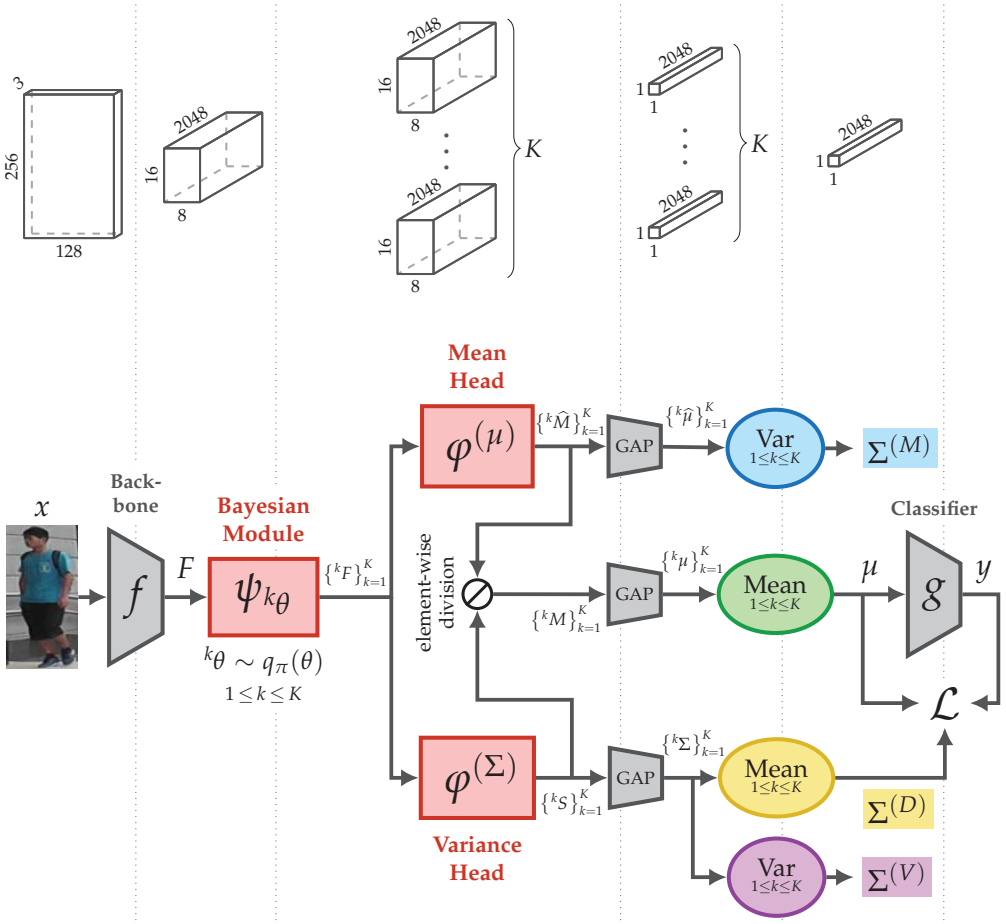


Figure 3. Overview of our architecture to estimate the three types of uncertainty for an embedding. First, the backbone f produces a feature tensor F from the input x. Next, the parameters ${}^k\theta$ of the Bayesian module are sampled K times from the weight distribution $q_\pi(\theta)$, thus producing K versions of the Bayesian module, each of which processes the feature tensor F, producing a total of K resulting feature tensors $\left\{{}^kF\right\}_{k=1}^K$. Each of these resulting feature tensors ${}^kF$ is processed separately by the variance head $\varphi^{(\Sigma)}$, yielding the variance features ${}^kS$, and by the mean head $\varphi^{(\mu)}$, yielding the raw mean features ${}^k\widehat{M}$, which are divided element-wise by the variance features to obtain the mean features ${}^kM$. The mean and variance operations shown as ellipses are applied element-wise across the K versions of the raw mean vector ${}^k\widehat{\mu}$, the mean vector ${}^k\mu$, and the variance vector ${}^k\Sigma$, all of which are obtained through global average pooling (GAP) from their precursors. Computing the variance of the raw mean vector in this way yields the estimate for model uncertainty $\Sigma^{(M)}$. The mean of the mean vector yields the feature vector μ. The mean of the variance vector yields the estimate for data uncertainty $\Sigma^{(D)}$. And finally, the *variance* of the *variance* vector yields the estimate for distributional uncertainty $\Sigma^{(V)}$. Together, the feature vector and the data uncertainty estimate define the embedding distribution $\mathcal{N}(\mu, \Sigma)$, where $\Sigma = \mathrm{diag}(\Sigma^{(D)})$. During training, we additionally need a classifier g to map the feature vector μ to a categorical output y, which encodes different individuals in the training dataset. The feature vector μ, the data uncertainty estimate $\Sigma^{(D)}$, and the logits y, all contribute to the Loss $\mathcal{L}$. The shape of all named Tensors is sketched above the architecture for the respective sections. Mathematical and algorithmic details are explained in Section 3.3.

3.3.1. Bayesian Module

The first of these modules is the Bayesian module $\psi_{k\theta}$, which acts as a pre-processing unit for the two embedding heads and enables the estimation of model uncertainty in the original approach as well as the estimation of distributional uncertainty with our

modification. It is essentially a three-layered bottleneck block (Figure 2a), whose weights ${}^k\theta$ are sampled K times for each input, as described in Algorithm 1. The resulting K versions of the module each process the output of the backbone F to compute a feature tensor ${}^kF \in \mathbb{R}^{d\times 16\times 8}$ of the same size (Equation (4)).

$$ {}^kF = \psi_{k_\theta}(F), \quad k \in \{1, \ldots, K\} \tag{4} $$

The weight sampling can be more easily understood as dropping each entry of the full weight tensors $\widehat{\theta}$, which are learned by ordinary backpropagation, to 0 with probability $p = 0.3$, which is known as DropConnect. The number of weight samples K is 1 during training but is increased to 30 during inference which then allows the uncertainty estimation. This is identical to the hyperparameters used by Dou et al. [25].

Algorithm 1: Bayesian Module

Input: Extracted feature tensor F according to Equation (3)
Weight tensors $\widehat{\theta}$
Number of weight samples K
Drop-probability p for DropConnect

Output: Output features of the Bayesian module $\left\{ {}^kF \right\}_{k=1}^{K}$

for $k \in \{1, \ldots, K\}$ **do**
  ${}^k\theta \leftarrow \widehat{\theta}$ `// Initialize` ${}^k\theta$ `with weights of Bayesian module`
  `//` ${}^k\theta$ `contains the weights of several tensors`
  `// but is viewed as vector here for simplicity`
  for ${}^k\theta_i \in {}^k\theta$ **do**
    Sample $\beta \sim \text{Bernoulli}(1-p)$
    ${}^k\theta_i \leftarrow \beta \cdot {}^k\theta_i$ `// Apply DropConnect`
  ${}^kF \leftarrow \psi_{k_\theta}(F)$ `// Apply Bayesian module using sampled weights`
return $\left\{ {}^kF \right\}_{k=1}^{K}$

Discussion of the Number of Weight Samples during Training

In preliminary experiments, we investigated the effect of using more than one weight sample during both training and inference. In this case, we consistently observed a small decrease in the re-identification quality.

Why is a single weight sample beneficial during training? In the case of $K = 1$ during training, the model is required to produce an accurate embedding distribution based on the single weight configuration it has access to. The model must learn to produce these, for any given weight perturbation it may encounter. However, in the case of $K > 1$ during training, the prediction of the embedding distribution results from the average over the output for all K weight configurations. There is no requirement that any individual output is perfect or fit any particular weight configuration perfectly. It is sufficient for the resulting average to be a good prediction. Consequently, the quality of the output for each single weight configuration can decline, or become less well-adapted.

3.3.2. Embedding Heads

Like the Bayesian module, the mean and variance heads $\varphi^{(\mu)}, \varphi^{(\Sigma)}$ are also bottleneck-blocks but without any weight-sampling (see Figure 2b,c). They are both applied to each of the K versions of the resulting feature tensor ${}^kF$ separately which then consequently results

in K versions of the raw mean features ${}^{k}\widehat{M} \in \mathbb{R}^{d\times 16\times 8}$ (Equation (5)) and the variance features ${}^{k}S \in \mathbb{R}^{d\times 16\times 8}$ (Equation (6)).

$$ {}^{k}\widehat{M} = \varphi^{(\mu)}\left({}^{k}F\right) \tag{5} $$

$$ {}^{k}S = \varphi^{(\Sigma)}\left({}^{k}F\right) \tag{6} $$

The raw mean features ${}^{k}\widehat{M}$ and the variance features ${}^{k}S$ are then utilized according to Equation (7) by applying element-wise division ($\oslash$) to obtain the mean features ${}^{k}M \in \mathbb{R}^{d\times 16\times 8}$. This represents the first manner in which our approach employs the uncertainty values to enhance the embedding. In contrast to most approaches, this application of the data uncertainty estimate is also employed during inference.

$$ {}^{k}M = {}^{k}\widehat{M} \oslash {}^{k}S \tag{7} $$

The subsequent Global Average Pooling (GAP) is applied to each of the K versions of the raw mean features ${}^{k}\widehat{M}$, the mean features ${}^{k}M$, and the variance features ${}^{k}S$ separately, collapsing the width and height dimensions. This results in K versions of each of the raw mean vector ${}^{k}\widehat{\mu} \in \mathbb{R}^{d}$ (Equation (8)), the mean vector ${}^{k}\mu \in \mathbb{R}^{d}$ (Equation (9)), and the variance vector ${}^{k}\Sigma \in \mathbb{R}^{d}$ (Equation (10)).

$$ {}^{k}\widehat{\mu} = \mathrm{GAP}\left({}^{k}\widehat{M}\right) \tag{8} $$

$$ {}^{k}\mu = \mathrm{GAP}\left({}^{k}M\right) \tag{9} $$

$$ {}^{k}\Sigma = \mathrm{GAP}\left({}^{k}S\right) \tag{10} $$

In order to obtain the final mean vector $\mu \in \mathbb{R}^{d}$, the element-wise mean is computed across the K versions of the mean vector ${}^{k}\mu$ (Equation (11)). Here, $\mathbb{E}$ refers to the expected value of the indicated values.

$$ \mu_i = \mathbb{E}\left\{ {}^{k}\mu_i \right\}_{k=1}^{K} \tag{11} $$

By utilizing the Bayesian module and the embedding heads, the three types of uncertainty can be estimated, which will be explained in detail below.

3.3.3. Estimation of Model Uncertainty

The model uncertainty estimate $\Sigma^{(M)} \in \mathbb{R}^{d}$ is calculated as the element-wise variance over the K versions of the raw mean vector ${}^{k}\widehat{\mu}$ (Equation (12)). This is identical to the method proposed by Dou et al. [25].

$$ \Sigma_i^{(M)} = \mathbb{V}\mathrm{ar}\left\{ {}^{k}\widehat{\mu}_i \right\}_{k=1}^{K} \tag{12} $$

Here, $\mathbb{V}$ar refers to the variance of the indicated values. This formulation does not align particularly well with the mathematical theory in Equation (2). Most notably, this estimate is not independent of the input as the term $p(\theta \mid \mathcal{D})$ from Equation (2) would suggest the model uncertainty estimate should be. Instead, we propose that $\Sigma^{(M)}$ measures the effect of model uncertainty on the feature vector. A more detailed discussion is given in Section 3.4.

3.3.4. Estimation of Data Uncertainty

Data uncertainty is described in the mathematical model (Equation (2)) as a distribution $p(z \mid \xi)$ over the embedding space. As is common with probabilistic embeddings, our architecture approximates this distribution as a Gaussian $p(z \mid \xi) \approx \mathcal{N}(\mu, \Sigma)$ by estimating its parameters μ and Σ. The final data uncertainty estimate $\Sigma^{(D)} \in \mathbb{R}^{d}$ is obtained

by computing the element-wise mean across the K versions of the variance vector ${}^{k}\Sigma$ (Equation (13)).

$$\Sigma_i^{(D)} = \mathbb{E}\left\{ {}^{k}\Sigma_i \right\}_{k=1}^{K} \tag{13}$$

$$\Sigma = \operatorname{diag}\left(\Sigma^{(D)}\right) \tag{14}$$

The diagonal covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ of the multivariate normal distribution is obtained by using $\Sigma^{(D)}$ as the diagonal of the matrix (Equation (14)).

The reason for the model's ability to encode data uncertainty into the variance vector ${}^{k}\Sigma$ can be found in the loss function (Section 3.3.7). In essence, the model is encouraged to assign higher uncertainty values to instances where the classification is bad. During training, the model learns to distribute these higher values towards vector positions that are primarily responsible for the inferior classification outcome.

3.3.5. Estimation of Distributional Uncertainty

Our proposed distributional uncertainty estimate $\Sigma^{(V)} \in \mathbb{R}^d$ is calculated as the element-wise variance over the K versions of the variance vector ${}^{k}\Sigma$ (Equation (15)).

$$\Sigma_i^{(V)} = \mathbb{V}\mathrm{ar}\left\{ {}^{k}\Sigma_i \right\}_{k=1}^{K} \tag{15}$$

In the mathematical model (Equation (2)), distributional uncertainty resides in the distribution $p(\xi \mid x, \theta)$ over the embedding distribution $p(z \mid \xi)$. Since we approximated $p(z \mid \xi)$ as a Gaussian $\mathcal{N}(\mu, \Sigma)$ which is solely defined by its parameters, we can say that $\xi = (\mu, \Sigma)$ in our case. Hence, $p(\xi \mid x, \theta)$ becomes $p((\mu, \Sigma) \mid x, \theta)$.

The reason for excluding μ here and focusing only on $p(\Sigma \mid x, \theta)$ is that the variability in the raw mean vector $\widehat{\mu}$ is already utilized as model uncertainty (see Section 3.3.3). The mean vector μ is merely a result of the element-wise division of mean features by variance features (see Equation (7)), so considering it does not offer any new information. Furthermore, this approach ensures consistency in the shape of our uncertainty estimates as vectors of the same length as the embedding vector. A detailed investigation that supports the hypothesis that the vector $\Sigma^{(V)}$ actually measures distributional uncertainty is provided in Section 4.4.

3.3.6. Obtaining a Scalar Uncertainty Quantifier

In certain situations, it is very helpful to have a detailed uncertainty estimate which matches some other component's shape whose uncertainty it describes. We have already seen this in Equation (7) where we use the variance feature tensor to modify the mean feature tensor and will explore a similar strategy in Section 4.3. However, in other situations, it would be more helpful to have a simple and straightforward scalar uncertainty quantifier. This can be the case if the task is to define a cutoff for unacceptably high uncertainty or if the uncertainty values should be provided for human interpretation. Thus we need to choose a function to convert our uncertainty estimates $\Sigma^{(D)}, \Sigma^{(M)}, \Sigma^{(V)} \in \mathbb{R}^d$ to a scalar uncertainty score $s \in \mathbb{R}$.

Given that the data uncertainty estimate is employed as the diagonal of a covariance matrix in our approach and that the other two uncertainty vectors are also calculated as variances, it appears reasonable to consider the distributions to which these variances pertain and to utilize tools of probability theory in order to quantify the uncertainty inherent in these distributions. In particular, entropy is commonly used for this purpose. This is because the entropy H of some random variable A is defined as the expected surprisal. Naturally, we consider the entropy of the probabilistic embedding $\mathcal{N}(\mu, \operatorname{diag}(\Sigma^{(D)}))$ for data uncertainty. Analogously, we consider the entropy of $\mathcal{N}(\widehat{\mu}, \operatorname{diag}(\Sigma^{(M)}))$ and $\mathcal{N}(\Sigma^{(D)}, \operatorname{diag}(\Sigma^{(V)}))$ for model and distributional uncertainty. The entropy of a multivariate normal distribution is given in Equation (16).

$$H(A) = \frac{d}{2}\ln(2\pi e) + \frac{1}{2}\ln\det\Sigma, \qquad A \sim \mathcal{N}(\mu, \Sigma) \tag{16}$$

Because this formula only depends on Σ, and because we are working with diagonal covariance matrices whose diagonals are the uncertainty vectors, this simplifies to Equation (17) which describes the function $\mathcal{H} : \mathbb{R}^d \to \mathbb{R}$ that maps the uncertainty vector to the entropy of the respective distribution.

$$\mathcal{H}\left(\Sigma^{(\omega)}\right) = \frac{d}{2}\ln(2\pi e) + \frac{1}{2}\sum_{i=1}^{d}\ln\left(\Sigma_i^{(\omega)}\right), \qquad \omega \in \{M, D, V\} \tag{17}$$

We will use this measure in our experiments where a scalar uncertainty value is needed, i.e., for correlation analyses (Sections 4.2 and 4.3.2) and for the utilization of uncertainty to predict the extent to which an input is out-of-distribution (Section 4.4).

3.3.7. Loss

As detailed in Equation (18), the loss function consists of the classification and data uncertainty loss $\mathcal{L}_{\text{C\&DU}}$, Triplet Hard Loss $\mathcal{L}_{\text{TriH}}$ [67], and a weight decay term $\lambda \cdot \mathcal{L}_2^{\text{reg}}$.

$$\mathcal{L} = \mathcal{L}_{\text{C\&DU}} + \mathcal{L}_{\text{TriH}} + \lambda \cdot \mathcal{L}_2^{\text{reg}} \tag{18}$$

$$\mathcal{L}_{\text{C\&DU}} = \frac{1}{\sigma} \cdot \mathcal{L}_{\text{CE}} + \ln\sigma, \qquad \text{with } \sigma = \frac{1}{d}\left\|\Sigma^{(D)}\right\|_1 \tag{19}$$

The classification and data uncertainty loss $\mathcal{L}_{\text{C\&DU}}$, as defined in Equation (19), is essentially a weighted cross entropy loss ($\mathcal{L}_{\text{CE}}$). The size of the logit vector $y = g(\mu)$, $y \in \mathbb{R}^T$, which is needed to compute the cross entropy loss $\mathcal{L}_{\text{CE}}$, depends on the number T of individuals in the training dataset.

The cross-entropy loss is multiplied by the inverse of the data uncertainty term ($\frac{1}{\sigma} \cdot \mathcal{L}_{\text{CE}}$). As very small uncertainty values would result in an increased loss, the model is forced to learn appropriate uncertainty values greater than zero. Furthermore, large uncertainty values are encouraged in cases where the classification is poor. As a consequence, a trivial solution to minimize this part of the loss would be reached by assigning very large uncertainty values regardless of the input. Thus, we employ the logarithmic addition of the uncertainty term σ as a penalizing term.

The application of weight decay and batch normalization layers supports this limiting influence by ensuring that weights and activations remain relatively small, including the activations utilized as uncertainty values.

In addition to these effects, which only regulate the magnitude of the uncertainty values, the information propagated back from the cross entropy and triplet loss terms ensure that the uncertainty values are relevant to the feature vector entries of the same position and work as correction terms utilized via element-wise division as intended. This is because the embedding vector that serves as a basis for these loss terms is itself a result of said element-wise division.

The classification and data uncertainty loss $\mathcal{L}_{\text{C\&DU}}$ represents the second way in which our approach utilizes the uncertainty values to improve the embedding. The third and final application of the estimated uncertainty values will be presented in Section 4.3.4.

3.3.8. Training

The objective of person re-identification (ReID) is to train a model that enables the comparison of images depicting a single person each, with the goal of identifying images that depict the same subject. Each image is cropped around the subject, such that the subject is centered in the image. The comparison is conducted based on a learned feature embedding, such that each input image x can be described by a d-dimensional feature vector $z \in \mathbb{R}^d$.

Learning a Feature Embedding

As described in Figure 3 above, in our architecture, we compute the feature vector $z = \mu$. In the training phase, batch normalization followed by a fully connected layer and a Softmax function are applied to the feature vector z. These serve as a classification layer, and thus map the feature vector z to a classification output y. For the purpose of the classification task, every person in the training dataset is treated as a single category. Cross-entropy serves as classification loss. During the training phase, batches are sampled such that they contain several images per individual. Therefore, each training batch contains both matching and non-matching image pairs. Given that the classification layer comprises a limited number of parameters, the model is compelled to learn a robust feature embedding to facilitate accurate classification. In addition to the classification task, triplets comprising an anchor image, a matching image, and a non-matching image present within the training batch are employed as input for the triplet loss function. All other loss terms in Equation (18) serve the purpose of forcing the model to learn adequate uncertainties. After training, the classification layer is no longer required and thus discarded.

Benchmarking

For benchmarking, a set of Q query images $\mathcal{X}^{\mathcal{Q}}$ (Equation (20)) is compared to a set of G gallery images $\mathcal{X}^{\mathcal{G}}$ (Equation (21)).

$$\mathcal{X}^{\mathcal{Q}} = \left\{ x_1^{\mathcal{Q}}, x_2^{\mathcal{Q}}, \ldots, x_Q^{\mathcal{Q}} \right\} \tag{20}$$

$$\mathcal{X}^{\mathcal{G}} = \left\{ x_1^{\mathcal{G}}, x_2^{\mathcal{G}}, \ldots, x_G^{\mathcal{G}} \right\} \tag{21}$$

The objective of ReID is to identify those gallery images that depict the same individual as each of the query images. Therefore, each image $x_i^{\mathcal{Q}}$ of the query set $\mathcal{X}^{\mathcal{Q}}$ is described by a feature vector $z_i^{\mathcal{Q}} \in \mathbb{R}^d$. All of the query feature vectors are then combined to form a matrix $Z^{\mathcal{Q}} \in \mathbb{R}^{Q \times d}$ (Equation (22)). Similarly, the gallery images $\mathcal{X}^{\mathcal{G}}$ are each described by a feature vector $z_i^{\mathcal{G}} \in \mathbb{R}^d$. All of the gallery feature vectors are then combined to form a matrix $Z^{\mathcal{G}} \in \mathbb{R}^{G \times d}$ (Equation (23)).

$$Z^{\mathcal{Q}} = \begin{bmatrix} z_1^{\mathcal{Q}} & z_2^{\mathcal{Q}} & \ldots & z_Q^{\mathcal{Q}} \end{bmatrix}^T \tag{22}$$

$$Z^{\mathcal{G}} = \begin{bmatrix} z_1^{\mathcal{G}} & z_2^{\mathcal{G}} & \ldots & z_G^{\mathcal{G}} \end{bmatrix}^T \tag{23}$$

Based on the query features $Z^{\mathcal{Q}}$ and gallery features $Z^{\mathcal{G}}$, a distance matrix $D \in \mathbb{R}^{Q \times G}$ is computed. This is accomplished by applying a given distance function $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, which can be used to compare two vectors (Equation (24)). In our case, we use the cosine distance δ_{cos} (Equation (25)).

$$D_{i,j}(Z^{\mathcal{Q}}, Z^{\mathcal{G}}) = \delta(z_i^{\mathcal{Q}}, z_j^{\mathcal{G}}) \tag{24}$$

$$\delta_{cos}(z_i^{\mathcal{Q}}, z_j^{\mathcal{G}}) = 1 - \frac{z_i^{\mathcal{Q}} \cdot z_j^{\mathcal{G}}}{\left\| z_i^{\mathcal{Q}} \right\|_2 \cdot \left\| z_j^{\mathcal{G}} \right\|_2} \tag{25}$$

For the sake of simplicity, we define the operator $\mathfrak{d}_{cos} : \mathbb{R}^{Q \times d} \times \mathbb{R}^{G \times d} \to \mathbb{R}^{Q \times G}$ such that it compares all query feature vectors $Z^{\mathcal{Q}}$ and gallery feature vectors $Z^{\mathcal{G}}$ to compute the distance matrix D using the cosine distance δ_{cos} for comparing the feature vectors.

The distance matrix D is then used to compute rankings for each query. These rankings serve as the basis for evaluating the quality of the re-identification approach.

3.4. Alignment of the Estimation of the Three Uncertainty Types with the Mathematical Theory

Above, we described the mathematical theory for three types of uncertainty and their estimation in our architecture. Below, we discuss how well the estimated uncertainty vectors align with the mathematical theory.

3.4.1. Model Uncertainty

Whether or not the estimated uncertainty vector $\Sigma^{(M)}$ is an appropriate measure of model uncertainty is debatable. According to the model uncertainty term in Equation (2), model uncertainty is only conditioned upon the training dataset. Thus, it should be data-independent.

On the one hand, in our architecture, we follow the interpretation of [25], who refer to the estimated vector $\Sigma^{(M)}$ as the model uncertainty. The estimated uncertainty vector $\Sigma^{(M)}$ is a measure of the variability introduced by the Bayesian Module and is mostly independent of the data uncertainty branch of the network. Therefore, it captures variability information that is independent of the other two uncertainty estimates. This is somewhat in accordance with the mathematical model.

On the other hand, the estimated uncertainty vector $\Sigma^{(M)}$ changes with each input during inference, although the model stays the same. Certainly, this does not align with the mathematical term in Equation (2).

Therefore, we should rather interpret the estimated uncertainty vector $\Sigma^{(M)}$ as the influence of model uncertainty on the current feature vector, which depends on the current input data. In our experiments, we show, that this estimate $\Sigma^{(M)}$ is an important piece of information that can be utilized to improve the feature vector.

3.4.2. Data Uncertainty

The estimation of data uncertainty in our architecture closely aligns with the mathematical theory. The loss function forces the model to learn to estimate when predictions are uncertain based on the current input data.

3.4.3. Distributional Uncertainty

The estimation of distributional uncertainty in our architecture also closely aligns with the mathematical theory.

We compute the distributional uncertainty estimate as the variance of the data uncertainty estimate. However, we could also interpret the distributional uncertainty term in Equation (2) as the distribution over the embedding distribution. Since we have a probabilistic embedding, it could be interpreted as the distribution over both the feature vector μ and the variance vector $\Sigma^{(D)}$. Following this interpretation, both $\Sigma^{(M)}$ and $\Sigma^{(V)}$ should be considered part of the distributional uncertainty estimate.

In our experiments, we show, that $\Sigma^{(V)}$ computed as the variance of the variance vector can be considered a good estimate of distributional uncertainty. This hypothesis is supported by the observation that the estimated distributional uncertainty $\Sigma^{(V)}$ is a very good fit to predict which data are out-of-distribution, which is the described property of distributional uncertainty in literature [7,26,27]. In contrast, the estimated uncertainty vector $\Sigma^{(M)}$, which we consider the model uncertainty estimate, is much less suitable to predict which data are out-of-distribution.

4. Experiments

To explore the utilization of estimated uncertainties, we conducted several experiments. In Section 4.1, we describe the experimental setup. In Section 4.2, we examine the correlation between uncertainty estimates of the three different uncertainty types. In Section 4.3, we explore the utilization of estimated model uncertainty to improve the feature vector. And finally, in Section 4.4, we explore the utilization of estimated distributional uncertainty to predict the extent to which an input image is out-of-distribution.

4.1. Experimental Setup

In order to benchmark our model and the estimated uncertainties, we decided in favor of person re-identification as the application. This application is a common choice in the state of the art on uncertainty estimation for feature vectors [22,23,28]. Person re-identification provides well-defined benchmarks while the task is very challenging. To enable a comparison of our results with multiple state-of-the-art approaches, we utilize the Market-1501 dataset [68] (see Section 4.1.1), which is the most frequently used within this context.

We report the mean average precision (mAP) and the rank-1 statistic of the cumulative match characteristic (CMC) curve, which are the most commonly utilized measures of re-identification quality [68]. To get meaningful results, we repeated all experiments 20 times. Therefore, we report the mean and standard deviation for each quality measure. It is important to note that the rank-1 statistic is primarily designed to assess the re-identification ability to identify the best matches to each query in the gallery, which are easiest to find. In contrast, the mAP assesses the quality of the entire ranking. Given that uncertainties contribute to enhanced performance in challenging cases, we anticipate that the mAP will prove a more effective measure. Consequently, the mAP results will be the primary focus of discussion.

In some cases, we need to select a specific model for visualization purposes. In these cases, out of the 20 trained models, we decided in favor of the model whose performance closely matches the mAP reported by Dou et al. [25], which is the basis for our architecture. The mAP reported by Dou et al. [25] is 87.0, while our selected model achieves an mAP of 87.03 when the re-identification is conducted identically to the UAL approach [25].

4.1.1. Dataset

We utilize the Market-1501 dataset [68] for the training and evaluation of our models. It is commonly employed in the state of the art as a benchmark for the person re-identification task. The Market-1501 dataset contains 12,936 training images of $T = 751$ individuals. Its test partition contains 750 individuals and is split into $Q = 3368$ query images and G = 15,913 gallery images. The training set and the query set consist of images with a width-to-height ratio of 1:2, which have been cropped tightly around the depicted individual. Each image depicts a person centered within the frame. Most gallery images also follow this configuration. However, the gallery set also contains 2798 distractor images that deviate from this configuration. The distractors are further analyzed in Section 4.4.

4.1.2. Implementation Details

The basis for our implementation is the Bag of Tricks (BOT) approach [66] implemented in the framework FastReID [69]. BOT uses a slightly modified ResNet-50 [70] and several data augmentation techniques to establish a strong baseline for person re-identification. The modifications include the removal of reduction of resolution at one place in the ResNet and the addition of batch normalization between the extracted feature vector and the classification layer during training. The data augmentation techniques include scaling of the input images to a resolution of 256×128 pixels, Random Erasing [71] with probability $p = 0.5$, random horizontal flipping with probability $p = 0.5$, and padding by 10 pixels followed by random cropping of size 256×128. As in the UAL approach [25], which serves as the basis for our architecture, we employ the modified ResNet model as the backbone of our architecture and also include the batch normalization before the classification layer. As a result of this choice of backbone, the dimensionality of the feature vectors, as well as the uncertainty vectors, is $d = 2048$.

The training pipeline follows the one of the UAL approach [25]. We train our model for 120 epochs by utilizing the Adam optimizer [72] with a basic learning rate of $\eta = 0.00035$. The learning rate scheduling includes a warm-up phase for 2000 iterations, which are approximately 10 epochs, with a linear increase from 0.1η to η. Furthermore, the learning rate is reduced by a factor of 0.1 at epochs 40 and 90. The training batches are constructed by

first selecting eight individuals at random and then four random images for each individual. Therefore, the batch size is 64. Our loss consists of the classification and data uncertainty loss $\mathcal{L}_{\text{C\&DU}}$ and the Triplet Hard Loss $\mathcal{L}_{\text{TriH}}$ with a margin of 0.3. As regularization, we use weight decay with a strength of $\lambda = 0.0005$ and label smoothing with a smoothing factor $\alpha = 0.1$.

For further implementation details, we refer to the source code that is publicly available Source code available at https://www.tu-ilmenau.de/neurob/data-sets-code/uncertainties-for-embeddings (accessed on 22 August 2024).

4.2. Analysis of Correlation between Uncertainty Types

In this section, we examine the degree of independence between the three types of estimated uncertainty. This should determine whether the uncertainty types are correlated, if they are conditioned upon each other, or whether they are independent due to the extraction at different positions in the neural network. If the latter were to be the case, this would be in contradiction with the assumption that the estimated uncertainties follow mathematical theory. In accordance with mathematical theory, we expect that two types of uncertainty estimates will be correlated to some degree if the term of one type of uncertainty in Equation (2) is conditioned upon the term of another uncertainty type. From Equation (2), we know that the distributional uncertainty term is conditioned upon the model uncertainty term, and the data uncertainty term is conditioned upon the distributional uncertainty term. Therefore, we expect that a correlation will be observed between the model uncertainty estimate and the distributional uncertainty estimate. Furthermore, we also expect that the data uncertainty estimate and the distributional uncertainty estimate will be correlated to some degree. Conversely, we expect only a low correlation between the model uncertainty estimate and the data uncertainty estimate.

In order to investigate these relationships, it is necessary to compare the three types of uncertainties for identical inputs. Accordingly, for the three types of uncertainties, we extracted estimated uncertainty vectors for the query and gallery partitions of the Market-1501 dataset [68]. Then, we derived scalar uncertainty values from the estimated uncertainty vectors by computing the entropy, as discussed in Section 3.3.6. Figure 4a–c shows pairwise joint distributions for each pair of uncertainty types. Furthermore, a linear fit was applied to each plot to generate a trend line, and the respective correlation is reported in the plots. The visualized contour lines of the joint distribution are generated using Kernel Density Estimation (KDE).

Additionally, Figure 4d shows the contour lines for different subsets of the Market-1501 dataset. The separability of these in-distribution and out-of-distribution sets by utilizing either data uncertainty or distributional uncertainty is investigated in Section 4.4.

As expected, Figure 4a illustrates that the model uncertainty estimate and the data uncertainty estimate only have a very low correlation. Furthermore, the outcome of the comparison between the distributional uncertainty estimate and the data uncertainty estimate, as illustrated in Figure 4b, aligns with our expectations. The same is true for the results of the comparison of the distributional uncertainty estimate and the data uncertainty estimate, as illustrated in Figure 4c. The correlation between the distributional uncertainty estimate and the data uncertainty estimate is 0.7485, while the correlation between the distributional uncertainty estimate and the model uncertainty estimate is 0.6938. Given that both correlation coefficients are around 0.7 or more, we can consider these pairs of uncertainty estimates to be highly correlated. Therefore, the results of the pairwise correlation analyses for the three types of estimated uncertainty provide support for the hypothesis that the estimated uncertainties are in accordance with the mathematical theory.

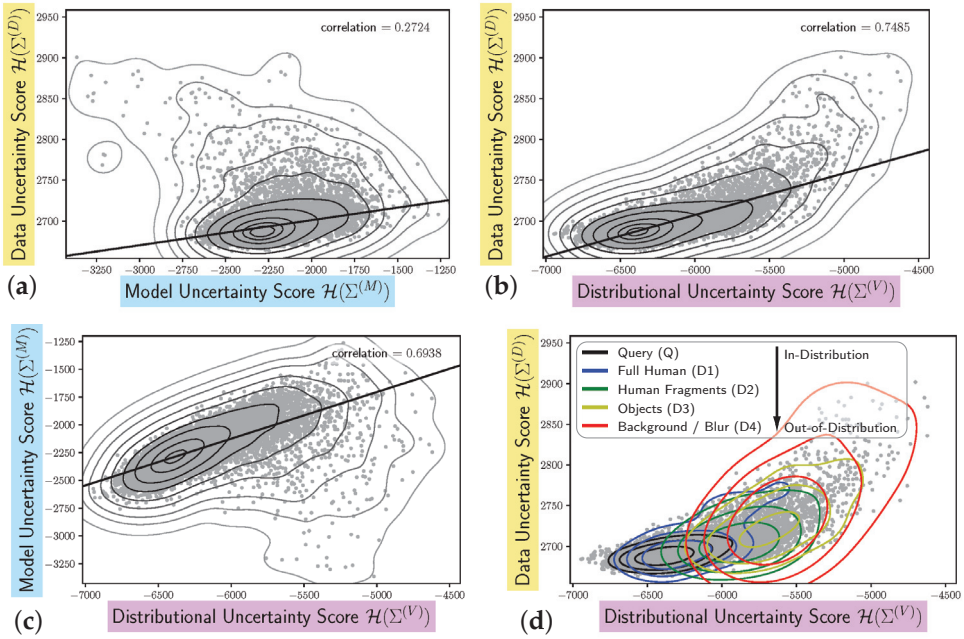


Figure 4. Correlation between the data, model, and distributional uncertainty estimates (**a**–**c**). Each point represents a pair of uncertainty estimates for a sample of the set of query and gallery images of the Market-1501 dataset [68]. Additionally, in (**d**), the contour lines of the distributions of the query and distractor sets are displayed. The distractor sets (D1)–(D4) are explained in Section 4.4.2. The colors are identical to those used in that section.

4.3. Utilization of Model Uncertainty during Inference

In this section, we examine how the model uncertainty estimate can be utilized during inference to refine the feature vector such that the re-identification quality improves. First, in Section 4.3.1, we report the results of related work and then rank our best result in comparison to them. Then, we derive step-by-step how model uncertainty should be utilized to refine the feature vector. Therefore, in Section 4.3.2, we derive why the model uncertainty estimate is suitable at all for refining the feature vector. In Section 4.3.3, we show that some simple solutions for incorporating model uncertainty into the process are not effective. In Section 4.3.4, we derive an appropriate technique for utilizing estimated model uncertainty in order to refine the feature vector, which consequently leads to an improved re-identification performance.

4.3.1. Comparison with the State of the Art

In Table 1, we list re-identification results of approaches that employ uncertainty estimation. The training and evaluation dataset was Market-1501 [68]. The results were either reported in related work [22,28,52] or are the results of our experiments.

It should be noted that the baseline (displayed in gray) in each of the three aforementioned publications differed. The results reported by Taha et al. [22] are based on the DenseNet baseline AWTL [73]. The results reported by An et al. [28] are based on a ResNet-50 baseline and the reported mean and standard deviation were calculated over 5 runs. The results reported by Zhang et al. [52] are based on the ResNet-50 baseline IDE [74]. It is therefore evident that attaining an improvement is considerably easier when one starts from a relatively weak baseline.

We follow the attempt of Dou et al. [25] and establish the strong BOT baseline [66] as a starting point. We report the average result and the standard deviation over 20 runs for all approaches with publicly available source code by the authors, namely Distribution Net (DistNet) [51], Probabilistic Face Embeddings (PFE) [21], and Uncertainty Aware Learning (UAL) [25]. For each approach, we list which types of uncertainty are estimated

during training and which types of uncertainties are utilized during inference for refining the feature vector. In our approach, we utilize the estimated model uncertainty $\Sigma^{(M)}$ and the estimated data uncertainty $\Sigma^{(D)}$ to refine the feature embedding vector μ. Therefore, we name our approach uncertainty-based embedding refinement (UBER). Furthermore, the incorporation of distributional uncertainty into the refinement of the feature embedding vector yields additional minor improvements (Section 4.3.4) and can be used for identifying input images that are out-of-distribution (Section 4.4).

Table 1. Comparison with state-of-the-art re-identification approaches that employ uncertainty estimation. The benchmarking dataset was Market-1501 [68]. For each approach, the types of uncertainties employed are listed.

Approach	Results Reported In	mAP [%]	rank-1 [%]	Uncertainty Training M	Training D	Training V	Inference M	Inference D	Inference V
AWTL baseline [73]	[22]	71.45	84.89	✗	✗	✗	✗	✗	✗
Bayesian Retrieval [22]	[22]	72.19	85.87	✓	✗	✗	✗	✗	✗
ResNet-50 baseline	[28]	75.65 ± 0.21	90.48 ± 0.45	✗	✗	✗	✗	✗	✗
DistNet [51]	[28]	74.61 ± 0.06	90.10 ± 0.29	✗	✓	✗	✗	✗	✗
PFE [21]	[28]	76.49 ± 0.23	90.48 ± 0.51	✗	✓	✗	✗	✗	✗
DUL [31]	[28]	77.12 ± 0.19	90.09 ± 0.17	✗	✓	✗	✗	✗	✗
MEIB [28]	[28]	79.67 ± 0.15	92.14 ± 0.16	✗	✓	✗	✗	✗	✗
IDE baseline [74]	[52]	78.4	89.7	✗	✗	✗	✗	✗	✗
LPP [52]	[52]	80.5	91.2	✗	✓	✗	✗	✗	✗
BOT baseline [66]		86.42 ± 0.09	94.53 ± 0.21	✗	✗	✗	✗	✗	✗
DistNet [51]		83.95 ± 0.20	93.44 ± 0.28	✗	✓	✗	✗	✗	✗
PFE [21]		86.42 ± 0.09	94.45 ± 0.20	✗	✓	✗	✗	✗	✗
UAL [25]		86.77 ± 0.14	94.64 ± 0.21	✓	✓	✗	✗	✓	✗
UBER (ours)		**87.21 ± 0.17**	**94.67 ± 0.18**	✓	✓	✓	✓	✓	✓

Additionally to the mean and standard deviation reported in Table 1, the results of all 20 runs of our experiments are shown in Figure 5 as box plots.

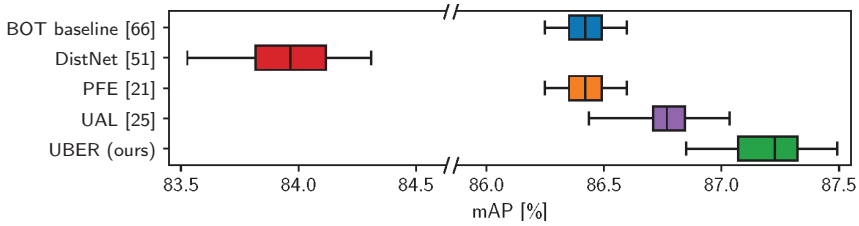


Figure 5. Comparison with state-of-the-art re-identification approaches that employ uncertainty estimation as box plots over the results of 20 runs based on the BOT baseline [66]. The benchmarking dataset was Market-1501 [68].

We can observe, that the performance of DistNet [51] declines significantly when it is applied to a strong baseline. This is consistent with the reported result by An et al. [28] (see Table 1). PFE [21] does not modify the feature vector during training. We observed that the learned uncertainty vector had very small values that barely affected the performance across various attempts to use them for embedding refinement. Therefore the result of PFE is basically identical to the baseline. UAL [25], which is the basis for our architecture, estimated uncertainties that helped to improve the performance over the strong baseline. The reported performance by Dou et al. [25] (mAP = 87.0, rank-1 = 95.2) could be reproduced in our experiments, although it is probably a cherry-picked result. Our approach UBER utilizes model uncertainty and data uncertainty. This helped to further improve the performance noticeably over the UAL result. 80% of our models trained in the

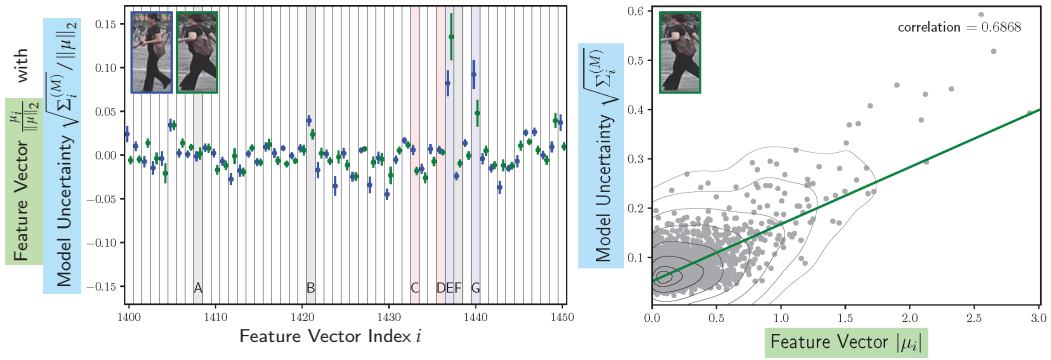
20 runs outperform the best of the 20 trained UAL models. The mean improvement of our approach UBER over UAL is higher than the improvement of UAL over the already strong baseline BOT. The highest mAP score achieved by the best of our trained UBER models across the 20 runs using an individually tailored constant c (see Section 4.3.4) was 87.52.

Below, a detailed explanation of the methodology employed to incorporate uncertainties into the refinement of the feature vector is provided, along with a justification for the selection of model uncertainty as the optimal type of uncertainty to be applied.

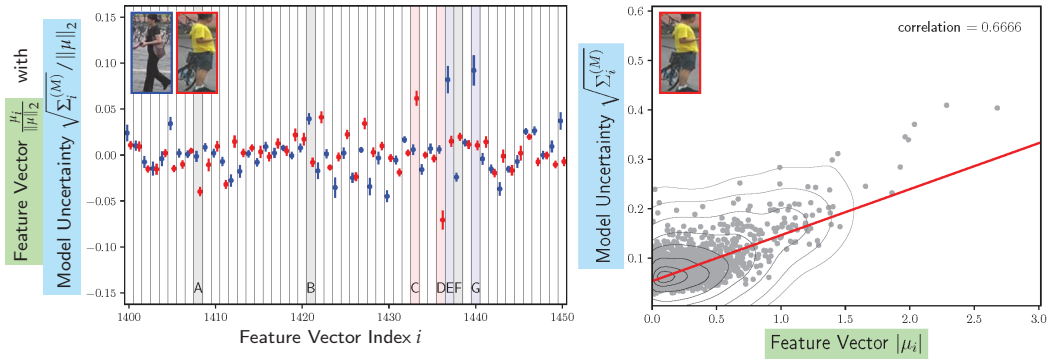
4.3.2. Relationship between the Feature Vector and Estimated Model Uncertainty

In order to explore the potential of incorporating estimated uncertainties during inference in order to improve the feature vectors, we investigated the relationship between the estimated uncertainty vectors and the feature vector. With regard to the model uncertainty estimate, we observed a correlation between the uncertainty values and the absolute values of the feature vector.

Figure 6a,c visualizes exemplary feature vectors and the associated model uncertainty values. Figure 6b,d shows the correlation.



(**a**) Matching Individual (**b**) Correlation



(**c**) Different Individuals (**d**) Correlation

Figure 6. Excerpt of the normalized feature vector (elements 1400–1450, $d = 2048$) for (**a**) two images of the same identity and (**c**) two images of different identities. The corresponding model uncertainty is shown as error bars representing the standard deviation. The highlighted positions marked as A–G are analyzed in the text. The correlation of the absolute values of the feature vector and the model uncertainty is shown for two input images in (**b**,**d**). The images are part of the Market-1501 dataset [68].

In Figure 6a,c an excerpt of feature vectors for three input images is presented. The dots represent the values of the normalized feature vector $\frac{\mu}{\|\mu\|_2}$, which is the basis for calculating the cosine distance. The error bars illustrate the estimated model uncertainty values as standard deviation.

In case of low estimated model uncertainty values for all three feature vectors, it can be observed that at specific positions, such as those highlighted as A, B, and F, we can see that the feature vectors exhibit relatively low differences for a matching individual (a)

and higher differences for different individuals (b). These differences are then aggregated, resulting in a low distance between the feature vectors of matching individuals and a higher distance between the feature vectors of different individuals.

However, the values of the feature vector at the highlighted positions E and G attain high absolute values for the images framed in blue and green of matching identity. Concurrently, the estimated model uncertainty is also relatively large at these locations. It is evident that there are large differences between the feature vectors of the two images depicting the same individual (a) at the highlighted positions. If the estimated model uncertainty is not taken into account, the differences would result in a greater distance between the feature vectors. Hence, the images would be evaluated as depicting individuals with notable dissimilarities.

For the feature vector corresponding to the non-matching image framed in red, we can also observe larger absolute values at the highlighted positions C and D. However, the estimated model uncertainty is much less pronounced than in the case described above.

Therefore, we hypothesize that the estimated model uncertainty is a valuable source of information to judge whether some outliers can be trusted or whether they are the result of uncertain model outputs.

Figure 6b,d illustrate the correlation between the elements of the estimated model uncertainty vector and the absolute values of the feature vector for two exemplary input images. Once more, the outliers can be primarily attributed to large model uncertainty. For the majority of feature vector elements, the estimated model uncertainty is relatively low.

4.3.3. Simple Solutions Prove to Be Ineffective

Before we describe the optimal method for integrating uncertainty information to enhance the re-identification performance, we first demonstrate that simple incorporation strategies are ineffective.

Replacement of the Feature Vector by the Estimated Uncertainty Vector

In light of the observed correlation in Figure 6b,d, one could hypothesize that the estimated model uncertainty may contain the same information as the feature vector. Therefore, we conducted an initial experiment to determine the extent to which re-identification results would be affected if the feature vector were replaced with the estimated model uncertainty vector. Furthermore, given that the three types of estimated uncertainty are also correlated (see Section 4.2), we also investigated the impact of replacing the feature vector with the estimated data uncertainty vector and the estimated distributional uncertainty vector.

For given query feature vectors $\mu^{\mathcal{Q}} \in \mathbb{R}^{Q \times d}$ and gallery feature vectors $\mu^{\mathcal{G}} \in \mathbb{R}^{G \times d}$, the conventional computation of the distance matrix D^{μ}, which serves as the basis for calculating the rankings and subsequently the re-identification performance, is described in Equation (26).

$$D^{\mu} = \mathfrak{d}_{cos}(\mu^{\mathcal{Q}}, \mu^{\mathcal{G}}) \tag{26}$$

This computation of the distance matrix can also be applied when the feature vector is replaced by an uncertainty vector. For given query uncertainty vector estimates $\Sigma^{(\omega),\mathcal{Q}} \in \mathbb{R}^{Q \times d}$, $\omega \in \{M, D, V\}$ and gallery uncertainty vector estimates $\Sigma^{(\omega),\mathcal{G}} \in \mathbb{R}^{G \times d}$, the distance matrix $D^{\Sigma^{(\omega)}}$ can be calculated as in Equation (27) with the variance used as the basis for the calculation. Similarly, the distance matrix $D^{\sqrt{\Sigma^{(\omega)}}}$, which employs the standard deviation as the basis, can be calculated as in Equation (28). In the case of the standard deviation, the square root is applied element-wise to the estimated uncertainty vectors.

$$D^{\Sigma^{(\omega)}} = \mathfrak{d}_{cos}(\Sigma^{(\omega),\mathcal{Q}}, \Sigma^{(\omega),\mathcal{G}}), \qquad \omega \in \{M, D, V\} \tag{27}$$

$$D^{\sqrt{\Sigma^{(\omega)}}} = \mathfrak{d}_{cos}\left(\sqrt{\Sigma^{(\omega),\mathcal{Q}}}, \sqrt{\Sigma^{(\omega),\mathcal{G}}}\right) \tag{28}$$

The results for replacing the feature vectors with the estimated uncertainty vectors are reported in Table 2.

Table 2. Validation: Can uncertainty vectors be utilized as a replacement for the feature vectors?

Distance Matrix		mAP [%]	rank-1 [%]
BOT baseline [66]		86.41 ± 0.09	94.42 ± 0.23
D^{μ} (identical to UAL [25])		86.77 ± 0.14	94.64 ± 0.21
$D^{\sqrt{\Sigma^{(M)}}}$	Uncertainty vector replaces feature vector	73.53 ± 0.52	91.05 ± 0.35
$D^{\Sigma^{(M)}}$		53.50 ± 1.42	78.55 ± 1.31
$D^{\sqrt{\Sigma^{(D)}}}$		41.25 ± 1.15	70.49 ± 1.05
$D^{\Sigma^{(D)}}$		41.17 ± 1.14	70.39 ± 1.08
$D^{\sqrt{\Sigma^{(V)}}}$		2.45 ± 0.45	2.79 ± 0.86
$D^{\Sigma^{(V)}}$		2.09 ± 0.36	2.21 ± 0.68

We observe that when the standard deviation of the model uncertainty estimate was utilized to replace the feature vector ($D^{\sqrt{\Sigma^{(M)}}}$), we unexpectedly obtained a relatively good result. Similarly, when the feature vector was replaced with the variance of the model uncertainty estimate ($D^{\Sigma^{(M)}}$) or the standard deviation or variance of the data uncertainty estimate ($D^{\sqrt{\Sigma^{(D)}}}$, $D^{\Sigma^{(D)}}$), the resulting outcome deviated significantly from the expected random outcome. In contrast, the distributional uncertainty estimates ($D^{\sqrt{\Sigma^{(V)}}}$, $D^{\Sigma^{(V)}}$) are unsuitable as a replacement. Thus, we can conclude, that the model uncertainty estimate and the data uncertainty estimate contain some information that is also present in the feature vector. However, even the best result achieved with the standard deviation of the model uncertainty estimate ($D^{\sqrt{\Sigma^{(M)}}}$) is significantly inferior to the results achieved with the feature vector (D^{μ}).

Simple Score-Level Fusion

In light of the unexpectedly good re-identification performance of the standard deviation of the model uncertainty estimate ($D^{\sqrt{\Sigma^{(M)}}}$) as a replacement for the feature vector, one could hypothesize that combining the re-identification results could yield further improvements. To validate this hypothesis, we conducted a second experiment to determine whether simple score-level fusion yields improvements. The distance scores for each pair of query and gallery images are stored in the distance matrices. In this case, the matrices are D^{μ} for the feature-vector-based scores and $D^{\sqrt{\Sigma^{(M)}}}$ for the uncertainty-based scores. Therefore, the distance matrices can be combined element-wise.

For score-level fusion, we considered an additive combination (Equation (29)) and a multiplicative combination (Equation (30)) of the distance scores, where $\odot$ symbolizes an element-wise multiplication.

$$D^{+M} = D^{\mu} + D^{\sqrt{\Sigma^{(M)}}} \tag{29}$$

$$D^{\odot M} = 1 - (1 - D^{\mu}) \odot \left(1 - D^{\sqrt{\Sigma^{(M)}}}\right) \tag{30}$$

Table 3 shows the score-level fusion results. The additive combination (D^{+M}) does not result in an improved re-identification performance. Furthermore, the multiplicative combination ($D^{\odot M}$) offers only a marginal improvement over the sole feature-vector-based re-identification (D^{μ}). Therefore, simple score-level fusion cannot achieve notable improvements. To this end, it is essential to identify a more effective approach to incorporate the model uncertainty estimate.

Table 3. Validation: Can simple score-level fusion of the model-uncertainty-based re-identification and feature-vector-based re-identification yield improvements?

Distance Matrix	mAP [%]	rank-1 [%]
BOT baseline [66]	86.41 ± 0.09	94.42 ± 0.23
D^{μ} (identical to UAL [25])	86.77 ± 0.14	94.64 ± 0.21
$D^{\odot M}$	86.80 ± 0.14	94.72 ± 0.18
D^{+M}	86.74 ± 0.14	94.73 ± 0.18

4.3.4. Utilizing the Estimated Model Uncertainty to Adjust Feature Vectors

Besides the combination of distance matrices, we can also consider adjusting the feature vector by utilizing the model uncertainty in order to achieve an improvement.

Adjusting the Feature Vector

Similarly to the application of data uncertainty in UAL [25], the feature vector can be divided element-wise by the estimated model uncertainty vector. If necessary, a constant c can be added to the uncertainty vector before it is applied to the feature vector. This is to circumvent the issue of division by small numbers. The adjusted feature vector $\bar{\mu}^{\mathcal{Q}/\mathcal{G}}$ with the model uncertainty employed as standard deviation $\sqrt{\Sigma^{(M),\mathcal{Q}/\mathcal{G}}}$ is computed as in Equation (31), where $\odot$ represents the element-wise multiplication. The distance matrix $D^{\mu,\sqrt{\Sigma^{(M)}}}$ is then computed using the adjusted query feature vectors $\bar{\mu}^{\mathcal{Q}}$ and gallery feature vectors $\bar{\mu}^{\mathcal{G}}$ as in Equation (32).

$$\bar{\mu}^{\mathcal{Q}/\mathcal{G}} = \mu^{\mathcal{Q}/\mathcal{G}} \odot \frac{1}{\sqrt{\Sigma^{(M),\mathcal{Q}/\mathcal{G}}} + c} \tag{31}$$

$$D^{\mu,\sqrt{\Sigma^{(M)}}} = \mathfrak{d}_{cos}(\bar{\mu}^{\mathcal{Q}}, \bar{\mu}^{\mathcal{G}}) \tag{32}$$

In order to find an appropriate value for the constant c, we performed a logarithmic hyperparameter search and selected the value of c that yielded the best average mAP across all 20 runs. The result achieved with the best identified hyperparameter $c = 0.2435$, is presented in Table 4. We also show the result when the estimated model uncertainty is employed as variance $D^{\mu,\Sigma^{(M)}}$, which was achieved with the best identified hyperparameter $c = 0.1225$.

Table 4. Validation: Can the estimated model uncertainty vector be used to adjust the feature vector?

Distance Matrix	mAP [%]	rank-1 [%]
BOT baseline [66]	86.41 ± 0.09	94.42 ± 0.23
D^{μ} (identical to UAL [25])	86.77 ± 0.14	94.64 ± 0.21
$D^{\mu,\Sigma^{(M)}}$	87.21 ± 0.17	94.70 ± 0.17
$D^{\mu,\sqrt{\Sigma^{(M)}}}$	87.20 ± 0.17	94.70 ± 0.21

By employing an appropriate c, identified through hyperparameter search, we observe a notable improvement over UAL [25] for both versions of the estimated model uncertainty used to adjust the feature vector. The performance gain measured by the mAP is 0.44 percentage points on average, which is a greater improvement than UAL [25] achieved over the BOT baseline [66] (0.36 percentage points).

Estimating the Hyperparameter c Based on the Uncertainty Vector

With the adjustment of the feature vector, we can get a notable improvement in re-identification performance. However, we must perform a hyperparameter search to find an appropriate constant c. Below, we describe how we can compute the value of c based on the uncertainty vector, in order to eliminate the necessity for a hyperparameter search.

Furthermore, this offers the advantage, that c is not required to be a constant, and can rather be computed individually for each input based on the estimated uncertainty.

The basic observation that led us towards deriving a formula for calculating c based on an uncertainty vector was that the best value $c = 0.2435$ for the standard deviation version of the estimated model uncertainty was approximately twice the best value $c = 0.1225$ for the variance version of the estimated model uncertainty. Also, if we optimize c specifically for a single trained model, we get a similar observation. We hypothesize that by fulfilling Equation (33), we may be able to derive a formula to compute c. First, we can write the uncertainty vectors in the detailed version, to get Equation (34). Now it is more easily observable that the square root is applied element-wise, as mentioned above.

$$f\left(\sqrt{\Sigma^{(M)}}\right) = 2 \cdot f\left(\Sigma^{(M)}\right) \tag{33}$$

$$f\left(\left[\sqrt{\Sigma_1^{(M)}}\ \sqrt{\Sigma_2^{(M)}}\ \dots\ \sqrt{\Sigma_d^{(M)}}\right]\right) = 2 \cdot f\left(\left[\Sigma_1^{(M)}\ \Sigma_2^{(M)}\ \dots\ \Sigma_d^{(M)}\right]\right) \tag{34}$$

A function that solves Equation (34) is shown in Equation (35).

$$f\left(\Sigma^{(M)}\right) = \frac{\lambda}{\left|\ln\left(\prod_{i=1}^{d} \Sigma_i^{(M)}\right)\right|} \tag{35}$$

The mathematical proof that the function described in Equation (35) does fulfill Equations (33) and (34), is given in Equations (36)–(43).

$$f\left(\sqrt{\Sigma^{(M)}}\right) = f\left(\left[\sqrt{\Sigma_1^{(M)}}\ \sqrt{\Sigma_2^{(M)}}\ \dots\ \sqrt{\Sigma_d^{(M)}}\right]\right) \tag{36}$$

$$= \frac{\lambda}{\left|\ln\left(\prod_{i=1}^{d} \sqrt{\Sigma_i^{(M)}}\right)\right|} \tag{37}$$

$$= \frac{\lambda}{\left|\ln\left(\prod_{i=1}^{d} \left(\Sigma_i^{(M)}\right)^{0.5}\right)\right|} \tag{38}$$

$$= \frac{\lambda}{\left|\ln\left(\left(\prod_{i=1}^{d} \Sigma_i^{(M)}\right)^{0.5}\right)\right|} \tag{39}$$

$$= \frac{\lambda}{\left|0.5 \cdot \ln\left(\prod_{i=1}^{d} \Sigma_i^{(M)}\right)\right|} \tag{40}$$

$$= \frac{\lambda}{0.5 \cdot \left|\ln\left(\prod_{i=1}^{d} \Sigma_i^{(M)}\right)\right|} \tag{41}$$

$$= 2 \cdot \frac{\lambda}{\left|\ln\left(\prod_{i=1}^{d} \Sigma_i^{(M)}\right)\right|} \tag{42}$$

$$= 2 \cdot f\left(\Sigma^{(M)}\right) \qquad \square \tag{43}$$

Now, we can reformulate Equation (35) as a function of a norm of an uncertainty vector (Equations (44)–(47)):

$$f\left(\Sigma^{(M)}\right) = \frac{\lambda}{\left|\ln\left(\prod_{i=1}^{d}\Sigma_i^{(M)}\right)\right|} \tag{44}$$

$$= \frac{\lambda}{\left|\sum_{i=1}^{d}\ln\left(\Sigma_i^{(M)}\right)\right|} \tag{45}$$

$$\begin{cases} = \frac{\lambda}{\sum_{i=1}^{d}\left|\ln\left(\Sigma_i^{(M)}\right)\right|} & \text{if } \Sigma_i^{(M)} < 1 \;\forall\; i \in \{1,2,\ldots,d\} \\ \approx \frac{\lambda}{\sum_{i=1}^{d}\left|\ln\left(\Sigma_i^{(M)}\right)\right|} & \text{otherwise} \quad \text{(in 2.88\% of uncertainty vectors)} \end{cases} \tag{46}$$

$$\begin{cases} = \frac{\lambda}{\left\|\ln\left(\Sigma^{(M)}\right)\right\|_1} & \text{if } \Sigma_i^{(M)} < 1 \;\forall\; i \in \{1,2,\ldots,d\} \\ \approx \frac{\lambda}{\left\|\ln\left(\Sigma^{(M)}\right)\right\|_1} & \text{otherwise} \quad \text{(in 2.88\% of uncertainty vectors)} \end{cases} \tag{47}$$

Note, that the derived formula using the norm (Equation (47)) is only equivalent to Equation (44) if all elements of the vector are smaller than 1. This requirement is fulfilled in the majority of cases. However, in 2.88% of the uncertainty vectors, at least one element is larger than 1. For these vectors, we performed a worst-case analysis, that showed, that the influence of the large values for some elements of the vector is marginal. Equation (48) describes the extent to which the two computations deviate in cases where $\Sigma_i^{(M)} > 1$ for some i. For the vector with the largest deviation over all runs, we observed a factor $\alpha = 1.0024$. This means that the error never exceeds 0.24%. This minimal error is negligible given that the achieved performance, as measured by mAP, is nearly unaffected when varying c by this amount.

$$\frac{\lambda}{\left\|\ln\left(\Sigma^{(M)}\right)\right\|_1} \leq \frac{\lambda}{\left|\sum_{i=1}^{d}\ln\left(\Sigma_i^{(M)}\right)\right|} < \alpha \cdot \frac{\lambda}{\left\|\ln\left(\Sigma^{(M)}\right)\right\|_1} \tag{48}$$

We conclude that we can use Equation (47) to compute c. Unfortunately, in Equation (47) a new parameter λ is introduced. Our hyperparameter search experiments show that while the best values for c deviate strongly for different trained models, they do not deviate as much if we use λ. The best value for λ is approximately half the dimensionality of the uncertainty vector $\frac{d}{2}$. Using $\lambda = \frac{d}{2}$, we can now compute an individual c for each input. Table 5 shows that applying this function to each input produces almost identical performance compared to the best results achieved with a constant c.

Table 5. Validation: Can the estimated model uncertainty vector be used to estimate the hyperparameter c for each input individually? To simplify the comparison to previous results, the results of Table 4 are repeated in the 3rd and 4th row.

Distance Matrix	mAP [%]	rank-1 [%]
BOT baseline [66]	86.41 ± 0.09	94.42 ± 0.23
D^{μ} (identical to UAL [25])	86.77 ± 0.14	94.64 ± 0.21
$D^{\mu,\Sigma^{(M)}}, c = 0.1225$	87.21 ± 0.17	94.70 ± 0.17
$D^{\mu,\sqrt{\Sigma^{(M)}}}, c = 0.2435$	87.20 ± 0.17	94.70 ± 0.21
$D^{\mu,\Sigma^{(M)}}, c = f\left(\Sigma^{(M)}\right)$	87.21 ± 0.18	94.68 ± 0.18
$D^{\mu,\sqrt{\Sigma^{(M)}}}, c = f\left(\sqrt{\Sigma^{(M)}}\right)$	87.19 ± 0.17	94.69 ± 0.20

With the approximation $\lambda = \frac{d}{2}$ and the application of Equation (47) for each input, we got almost identical results while removing the need for a hyperparameter search. We are now able to use the estimated model uncertainty to adjust the feature vector in a straightforward way.

Conclusion on the Utilization of Uncertainties

Data uncertainty is already utilized in our architecture to divide the result of the mean head by the result of the variance head, as in the UAL approach [25]. We showed that the estimated *model uncertainty* vector can be utilized to further adjust the feature vector.

The estimated distributional uncertainty is very helpful for identifying out-of-distribution data, which is explained in more detail in Section 4.4. Additionally, the estimated distributional uncertainty can be utilized to derive a scalar value to further improve the ReID performance, but only marginally, as explained below.

Further Investigations

To verify that our derived formula is a good way to estimate the hyperparameter c, we also experimented with a wide variety of possibilities. We examined the three types of uncertainties as the basis for the computation of a constant c, both as standard deviation and as variance. We derived a scalar value by applying the inverse of the l_1-norm of the logarithm ($(\|\ln(\cdot)\|_1)^{-1}$), the inverse of the l_1-norm ($(\|\cdot\|_1)^{-1}$), the entropy ($\mathcal{H}(\cdot)$), and the vector length ($\|\cdot\|_2$). For each of these scalar values, we performed a hyperparameter search to find an adequate multiplication factor λ. Furthermore, instead of the model uncertainty we also employed the other uncertainty types, both as standard deviation and as variance, to refine the feature vector as explained above. Overall, we checked 150 variations, each optimized via hyperparameter search. None of these variants improved the performance noticeably over the approach described above.

The best result is achieved by applying the inverse of the l_1-norm of the distributional uncertainty estimate as variance, and to refine the feature vector with the model uncertainty estimate as variance and the derived scalar constant. However, this result (mAP [%] = 87.213 ± 0.170) is only 0.007 percentage points better than the result when employing Equation (47) (87.206 ± 0.178). Therefore, it can be considered approximately equal. Nevertheless, the result described here, represents the best result achieved by our approach UBER in this paper. Therefore, this result is also shown in Table 1 and Figure 5 to compare with the state of the art. Specifically in Figure 5, we can see the significant improvement over UAL [25], which served as the basis for our network architecture.

Using the entropy, the inverse of the l_1-norm of the logarithm, or the inverse of the l_1-norm of any of the uncertainty types to derive the scalar value also achieved good results. The vector length was only appropriate in combination with the data uncertainty estimate.

4.4. Experimental Validation of the Distributional Uncertainty Estimate

As previously discussed in Section 3.1, distributional uncertainty is higher the more dissimilar the input is to the training data. The objective of this experiment is to verify that our distributional uncertainty estimate $\Sigma^{(V)}$ fits this description. Additionally, we test whether the other uncertainty estimates exhibit characteristics of a distributional uncertainty estimate. This is done to demonstrate that our distributional uncertainty estimate is the most appropriate for estimating distributional uncertainty. To that end, we construct four sets of out-of-distribution (OOD) images such that each successive set (D1)–(D4) represents a class of progressively greater dissimilarity to the training data (Section 4.4.1). We use the entropy function derived in Section 3.3.6 to translate our vector-based uncertainty estimates into scalar uncertainty scores and analyze the distribution of these scores for the different OOD sets in comparison to an in-distribution set (Section 4.4.2). Finally, we examine the score distribution of one uncertainty type in cases where another uncertainty type yields very low or very high scores (Section 4.4.3).

4.4.1. Out-of-Distribution Sets

The source of the images in these sets are the 2798 distractor images contained in the gallery partition of the Market-1501 dataset [68]. First, we manually categorized the distractors in detail and then summarized, arriving at four classes of similar difference to the training data. In this process, we also ensured that each category is composed of a reasonable number of images. Examples for the resulting categories can be seen in Figure 7 and a description of each set is provided below.

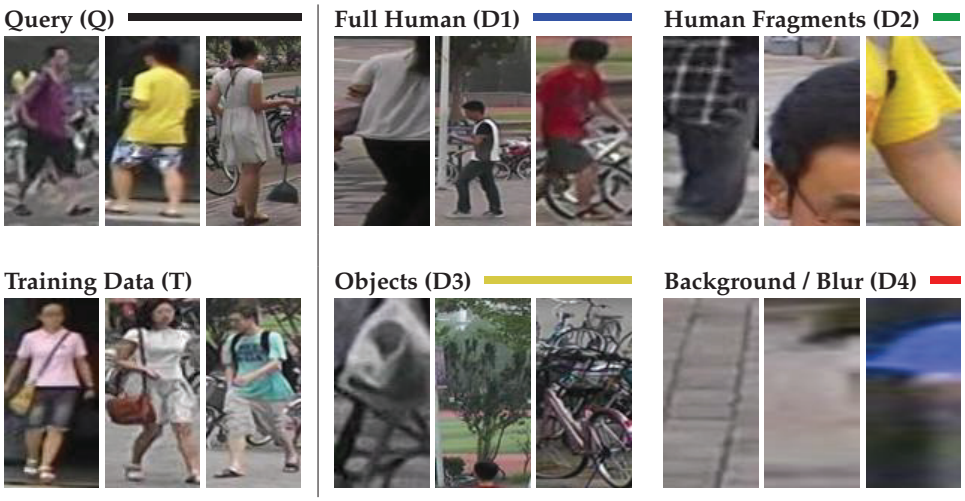


Figure 7. Examples of training images (T), the in-distribution set (Q), and the distractor sets (D1)–(D4). Images in (D1) contain large parts of the depicted humans. In (D2) they contain only fragments of a person. In (D3) they contain non-human objects. In (D4) they contain only background or blur. All images are part of the Market-1501 dataset [68].

Training Data (T) and Query (Q)

The images used during training (T) and as queries (Q) during inference both depict persons centered in the image. Since the training data (T) was already used for model training, we employ the query set (Q) as the in-distribution set for validation.

Full Human (D1)

The set (D1) contains images with imperfect bounding box placement, where the level of detail visible is nonetheless sufficient to enable a confident prediction of the identity depicted. Most of these images either have parts of the person missing (too zoomed-in) or show a lot of background (too zoomed-out). Some images exhibit bad image quality due to low resolution. This set is very similar to the training data. However, some relevant parts of the person might be missing and others are at different positions than expected.

Human Fragments (D2)

The set (D2) contains images that are zoomed-in to such an extent that only fragments of a person are visible which are not enough to confidently make a prediction of the identity depicted. Both the main content of the image and the background are known to the network but appear in different sizes, positions, and contexts in the training data.

Objects (D3)

The set (D3) contains images of non-human objects which can certainly be called a separate domain from what the model was trained on, even if it has seen them in the background of the training data. However, we still find the same general setup of a structured object placed in front of the known background, although now the foreground does not depict humans but objects such as bikes or trees.

Background/Blur (D4)

The set (D4) contains images where only background is visible or where only unrecognizable blobs of color can be seen that are caused by a combination of zoom and motion blur or focus blur. Here, not even the general setup remains the same: While the model may have seen some of the backgrounds in training images, it has not seen background without any foreground object. Furthermore, the blurred images are completely different from anything the model has seen. Both kinds of images do not contain the structured shapes the model expects.

In summary, the query set (Q) is used as the in-distribution set, while the distractor sets (D1)–(D4) represent out-of-distribution sets with increasing level of difference to the training set (T).

Labeling Process

For the labeling process, the distractor images were presented in random order and manually annotated according to the four categories. The labeling took 1–2 s per image. Due to the ambiguous nature of the images, imperfections in the composition of the distractor sets are to be expected. The number of images in the resulting sets (D1)-(D4) is given in Table 6. The labels for the distractors of the Market-1501 dataset are publicly available as part of the source code of this paper.

Table 6. Number of images in the training (T) and query (Q) set, as well as in the annotated subsets (D1)–(D4) of the distractors in Market-1501.

Set		**# of Images**
Training Data	(T)	12,936
Query	(Q)	3368
Full Human	(D1)	278
Human Fragments	(D2)	1729
Objects	(D3)	567
Background/Blur	(D4)	224

4.4.2. Analysis of Uncertainty Score Distributions

As mentioned above, the OOD sets (D1)–(D4) presented in the previous section represent classes of progressively greater dissimilarity to the training data. This means that the OOD degree, i.e., the amount to which an image deviates from the training distribution, increases from (D1) to (D4).

If our distributional uncertainty estimate $\Sigma^{(V)}$ is indeed a good estimate of distributional uncertainty, it should measure the OOD degree. Thus we expect to observe a gradual increase of the distributional uncertainty score from the distractor sets (D1)–(D4). Since the sets are not perfectly separable, an overlap between the distributions is to be expected. However, the distributions should be somewhat separable. At the very least, the means of the uncertainty score distributions for the sets (D1) to (D4) should appear in this order. Furthermore, we expect the distributional uncertainty estimate to outperform the other uncertainty estimates regarding the ability to separate the distributions.

Experimental Setup

As was laid out in detail in Section 3.3, the data uncertainty estimate $\Sigma^{(D)}$ is the diagonal of the covariance matrix of the probabilistic embedding, while the model and distributional uncertainty estimates $\Sigma^{(M)}$ and $\Sigma^{(V)}$ are the variance induced by the Bayesian module in the raw mean vector $\widehat{\mu}$ and data uncertainty estimate $\Sigma^{(D)}$.

All three uncertainty estimates are vectors, so it is not immediately clear how to calculate a scalar uncertainty score from them. As described in Section 3.3.6, we have decided in favor of entropy for this purpose.

After training our model on Market-1501, we generate the three uncertainty estimates for all relevant images regarding this experiment and compute the scalar uncertainty scores according to Equation (17) for each of them. Then, we employ kernel density estimation (KDE) to model the probability density functions corresponding to the observed values for each of the considered sets, i.e., the query set (Q) as the in-distribution set and the OOD sets (D1)–(D4). The resulting KDE plots are shown in Figure 8a–c.

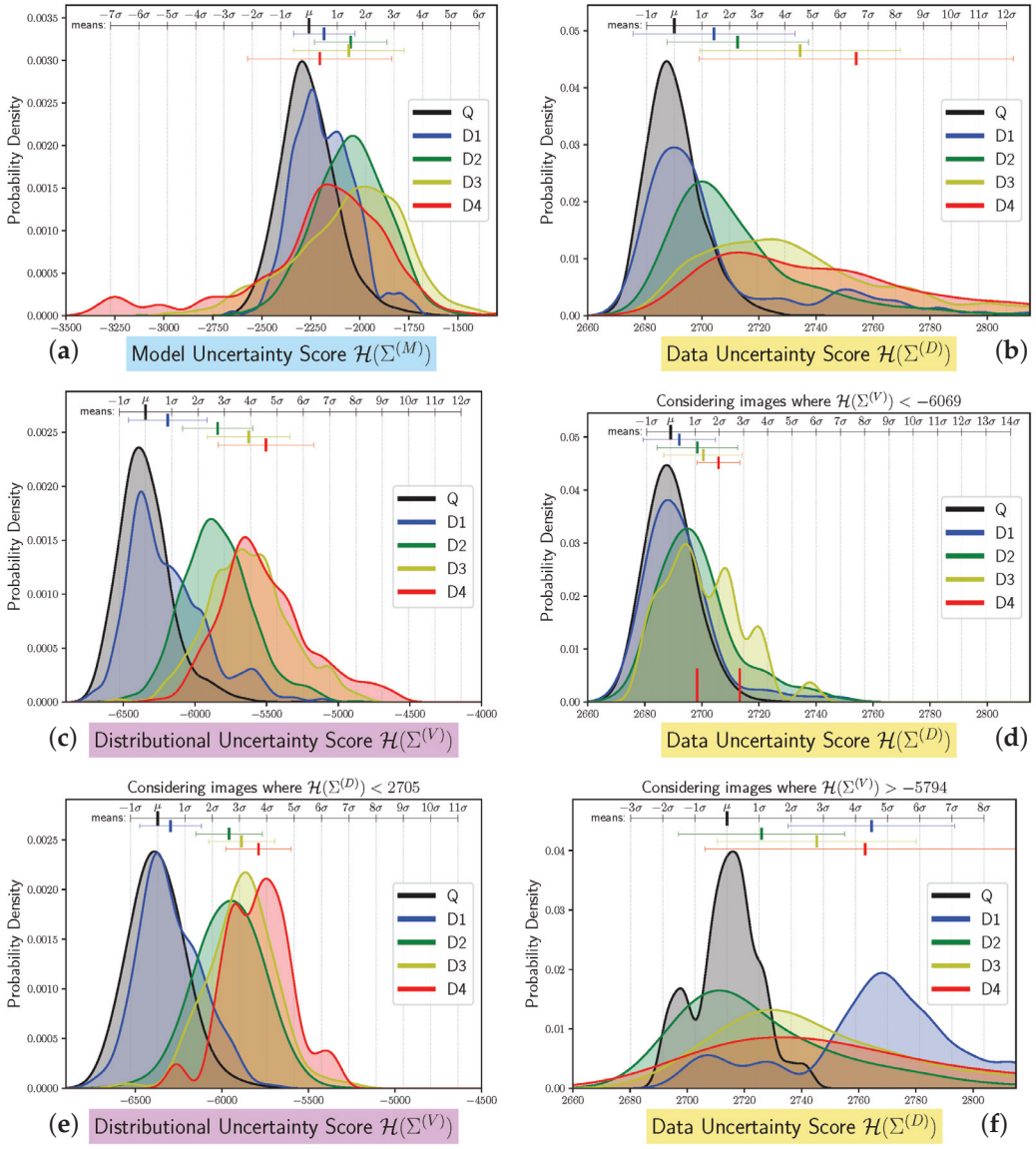


Figure 8. Distribution of uncertainty scores over all images in certain subsets of Market-1501 for the three uncertainty types (**a**) model uncertainty, (**b**) data uncertainty, and (**c**) distributional uncertainty. The five subsets (Q), (D1)–(D4) are described in Section 4.4.1 and examples are provided in Figure 7. (**d**) Data uncertainty in case of low distributional uncertainty. (**e**) Distributional uncertainty in case of low data uncertainty. (**f**) Data uncertainty in case of high distributional uncertainty. On top of each of the probability density plots, the means and standard deviations of each distribution are plotted in relation to the query set (Q).

Model Uncertainty Score Distribution

In the mathematical model (Equation (2)), the model uncertainty term $p(\theta \mid \mathcal{D})$ is both separate and independent of the distributional uncertainty term $p(\xi \mid x, \theta)$. Therefore, we would expect the model uncertainty score distributions for the different sets to be indifferent to the OOD degree, i.e., we should see no ordered separation of the sets.

Figure 8a shows the mean of the model uncertainty scores for all sets on top of the distributions. Table 7a lists how far away the mean of the uncertainty scores of the distractor sets (D1)–(D4) are from the in-distribution set (Q) in terms of the standard deviation of the in-distribution set. We can immediately see that the means of the distributions are not in the correct order. Furthermore, there exists a large overlap between the distributions of all sets. Surprisingly, some of the lowest model uncertainty values are observed for images in the distractor set (D4) with the highest OOD degree.

Table 7. Number of standard deviations of the query score distribution that the mean of the uncertainty score distribution of the respective set is above the mean of the query score distribution for (**a**) model uncertainty, (**b**) data uncertainty, (**c**) distributional uncertainty, (**d**) data uncertainty in case of low distributional uncertainty, (**e**) distributional uncertainty in case of low data uncertainty, and (**f**) data uncertainty in case of high distributional uncertainty. For a visualization, we refer to Figure 8.

Set		**(a)**	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**
Full Human	(D1)	0.53	1.43	0.85	0.35	0.47	4.50
Human Fragments	(D2)	1.47	2.29	2.74	1.10	2.61	1.07
Objects	(D3)	1.41	4.55	3.93	1.34	3.07	2.80
Background/Blur	(D4)	0.38	6.57	4.57	1.97	3.69	4.30

It appears that the model uncertainty estimate is unsuited to predicting the OOD degree. This fits our expectations based on the mathematical theory of how a model uncertainty estimate should behave in this situation.

Data Uncertainty Score Distribution

The data uncertainty term $p(z \mid \xi)$ in the mathematical model is separate from but not independent of the distributional uncertainty term $p(\xi \mid x, \theta)$. Ideally, the data uncertainty score distributions would be indifferent to the OOD degree, as it is a separate quantity. However, the conditional dependence of the data uncertainty term on the distributional uncertainty term likely means that this is not the case. As we saw in Figure 1, the reliability of the data uncertainty estimate decreases with increasing distributional uncertainty. Therefore, images with higher distributional uncertainty should exhibit larger spread in data uncertainty estimates around their actual data uncertainty.

As can be seen in Figure 8b and Table 7b, the means of the distributions are ordered by the OOD-degree of the respective set. In Figure 8b and Table 8b, we also observe that the uncertainty score distributions for the OOD sets exhibit standard deviations that are considerably higher than the one of the query set and are higher for greater OOD degree. Only the distractor set (D1) with the lowest OOD degree does not quite fit this pattern, since its standard deviation is larger than that of the distractor set (D2) with the slightly higher OOD degree.

Table 8. Factor by which the standard deviation of the uncertainty score distribution for the respective set is larger than the standard deviation of the query score distribution for (**a**) model uncertainty, (**b**) data uncertainty, (**c**) distributional uncertainty, (**d**) data uncertainty in case of low distributional uncertainty, (**e**) distributional uncertainty in case of low data uncertainty, and (**f**) data uncertainty in case of high distributional uncertainty. For a visualization, we refer to Figure 8.

Set		**(a)**	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**
Full Human	(D1)	1.08	2.92	1.50	1.49	1.13	2.60
Human Fragments	(D2)	1.28	2.55	1.33	1.66	1.21	2.59
Objects	(D3)	1.95	3.62	1.57	1.62	1.21	3.10
Background/Blur	(D4)	2.54	5.68	1.81	0.88	1.19	4.98

The data uncertainty estimate appears to be somewhat suitable for predicting the OOD degree, though the large standard deviations and largely overlapping distributions make it less than ideal. While this predictive ability seemingly contradicts the theoretical considerations, this may be due to the issue with its reliability in cases of high distributional uncertainty mentioned above. This is supported by the fact that the standard deviation of the data uncertainty score distributions generally increases with OOD degree, which is exactly what we expected from the mathematical model. While the distractor set (D1) with the lowest OOD degree does not quite fit this pattern, we can clearly see in Figure 8b that the main body of the distribution of (D1) does have a smaller standard deviation than that of the distractor set (D2), which has a slightly higher OOD degree. Furthermore, we can see that the unexpectedly high standard deviation of the distractor set (D1) is due to a smaller second mode in the distribution with images of higher data uncertainty scores. This is likely to be the part of the images in (D1) which show noticeably worse image quality compared to the others, which correctly is associated with a higher data uncertainty.

Overall, we have some supporting and some unclear evidence in relation to the theoretical considerations. Therefore, we also examine the distributions of only those images with low distributional uncertainty, which should be free of the observed effect. This investigation is further discussed in Section 4.4.3.

Distributional Uncertainty Score Distribution

Since distributional uncertainty is precisely the type of uncertainty that measures the OOD degree, we would expect a clear ordering of the distributional uncertainty score distributions and a clear distinction between the in-distribution set (Q) and the OOD sets (D1)–(D4). However, the distractor set (D1) with the lowest OOD degree is still largely similar to the in-distribution set (Q) and contains those images that still contain enough information to make a confident prediction regarding the identity. Therefore, we expect the distribution of the distractor set (D1) to be quite similar to that of the in-distribution set (Q). Conversely, the distractor set (D4) with the highest OOD degree should have a clear distinction from the in-distribution set (Q), even if it does not necessarily have to be perfectly separable due to the ambiguous nature of many images, as discussed in Section 4.4.1.

We can immediately see in Figure 8c and Table 7c that the means of the distributions are ordered by the OOD degree of the respective set. In Figure 8c and Table 8c, we can also compare the relative standard deviations of the distributions in terms of multiples of the standard deviation of the in-distribution set (Q) for the distractor sets (D1)–(D4). We observe that the relative standard deviations of the distributional uncertainty score distributions for the distractor sets (D1)–(D4) in Figure 8c and Table 8c are smaller than those of the data uncertainty score distributions for the distractor sets (D1)–(D4) in Figure 8b and Table 8b. Furthermore, we observe the same pattern of increase in standard deviation for an increase in OOD degree as for the data uncertainty score distributions, including the exception for the distractor set (D1). However, we observe much smaller relative standard deviations for the distributional uncertainty score distributions.

This seems to indicate a confident prediction of distributional uncertainty that does not become more unreliable with increased OOD degree of the distractor sets. The degree of overlap between the in-distribution set (Q) and the distractor set (D1) with the lowest OOD degree is very high. Therefore, the distractor set (D1) is rather judged as an in-distribution set, which is plausible considering that a re-identification of the depicted individuals is still possible. The degree of overlap between the in-distribution set (Q) and the respective distractor sets (D2)–(D4) is smaller the higher the OOD degree is. Therefore, the distributional uncertainty estimate seems to be a good predictor of the OOD degree. While the distributions are not perfectly separated, a cutoff can be chosen that excludes only very few images of the in-distribution set (Q) while still excluding the vast majority of the images in the distractor set (D4) with the highest OOD degree. Essentially, our observations perfectly matched our expectations from mathematical theory.

4.4.3. Uncertainty Score Distributions When Controlling for Another Uncertainty Type

Since our results regarding the prediction of the OOD degree by utilizing the data uncertainty estimate were somewhat inconclusive, in the following, we conduct further investigations. First, we examine the estimated data uncertainty for subsets of images for which low distributional uncertainty was estimated. Furthermore, we also examine the estimated distributional uncertainty for subsets of images for which low data uncertainty was estimated. This is intended to confirm that, as suggested by the mathematical model (Equation (2)), the dependency between data uncertainty and distributional uncertainty is unidirectional. Additionally, we examine the estimated data uncertainty for subsets of images for which high distributional uncertainty was estimated. This serves to verify the derivation from the mathematical model that given high distributional uncertainty, the data uncertainty estimate becomes unreliable.

Data Uncertainty in Case of Low Distributional Uncertainty

As discussed in Section 3.1.2, we should not trust the data uncertainty estimate in case of high distributional uncertainty. For this reason, we investigate the data uncertainty score distribution of images with low distributional uncertainty scores. Based on the mathematical theory, we expect that in case of low distributional uncertainty, the width of the data uncertainty score distributions should be small for all sets (Q), (D1)–(D4). Furthermore, we expect that a low distributional uncertainty score indicates a low OOD degree. Therefore, the ability of data uncertainty to appropriately order and separate the distractor sets should be strongly diminished.

In contrast to the previous experiments, we now do not compute the distributions on the entire sets (Q), (D1)–(D4), but instead filter the sets. First, we examine data uncertainty in case of low distributional uncertainty. Therefore, we use only the samples of the sets (Q), (D1)–(D4) that had a low distributional uncertainty score in Figure 8c. As criteria, we use a distance of 1.5 standard deviations with respect to the query set (Q) to the mean of the query set (Q), as defined in Equation (49).

$$\mathcal{H}\left(\Sigma^{(V)}\right) < \mathbb{E}\left\{\mathcal{H}\left(\Sigma_i^{(V),\mathcal{Q}}\right)\right\}_{i=1}^{Q} + 1.5 \cdot \sqrt{\mathbb{V}\mathrm{ar}\left\{\mathcal{H}\left(\Sigma_i^{(V),\mathcal{Q}}\right)\right\}_{i=1}^{Q}} \tag{49}$$

Therefore, the threshold for filtering the sets is $\tau_{\downarrow\mathcal{H}\left(\Sigma^{(V)}\right)} = -6069$. The remaining number of images in each set, along with the percentage of the original set size that remains, is provided in Table 9.

Table 9. Statistics for subsets of images with low distributional uncertainty ($\downarrow\mathcal{H}(\Sigma^{(V)})$).

Set		# of Images	Remaining
Query	$(\mathrm{Q})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$	3119	92.6%
Full Human	$(\mathrm{D1})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$	199	71.6%
Human Fragments	$(\mathrm{D2})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$	304	17.6%
Objects	$(\mathrm{D3})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$	31	5.5%
Background/Blur	$(\mathrm{D4})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$	2	0.9%

We can see, that only two images of the distractor set (D4) with the highest OOD degree remain. Similarly, only 31 images remain of the distractor set (D3) with the second-highest OOD degree.

Figure 8d shows the distributions of the reduced sets. Since the distractor set $(\mathrm{D4})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ contains only 2 samples, their uncertainty values are plotted instead of a distribution. We observe that the sets are still ordered correctly by the data uncertainty estimate. However, the distributions for the different sets share a large overlap. The distractor set $(\mathrm{D1})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ almost perfectly overlays the in-distribution set $(\mathrm{Q})^{\downarrow\mathcal{H}(\Sigma^{(V)})}$, with only some exceptions

that have a higher data uncertainty score. The distractor sets (D2)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ and (D3)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ with slightly higher OOD degree still share a large overlap with the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$. The two samples of the distractor set (D4)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ with the highest OOD degree also fall within the overlapping area of the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$.

Even though the means of the distributions of the sets remain in the correct order, the large overlap between the distributions of the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ and the distractor sets (D1)–(D4)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ shows that the data uncertainty estimate is not very useful for distinguishing the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ and the distractor sets (D1)–(D4)$^{\downarrow\mathcal{H}(\Sigma^{(V)})}$ in case of low distributional uncertainty. The reason that the data uncertainty estimate is still able to order the sets correctly to a certain extent can be attributed to three factors: (1) measurement errors and image corruptions, which contribute to both kinds of uncertainty; (2) labeling errors stemming from the ambiguous nature of the sets; and (3) the high correlation between data and distributional uncertainty, as discussed in Section 4.2.

Overall, we observe exactly what we can expect based on the mathematical theory. In case of low estimated distributional uncertainty, the distributions of the data uncertainty estimates are much sharper and the difference between the distractor sets is strongly diminished.

Distributional Uncertainty in Case of Low Data Uncertainty

Now, we investigate whether there are similar results for the distributional uncertainty score distributions when considering images with low data uncertainty scores. From Figure 4d, we can already observe that this is likely not the case. Instead, in case of low estimated data uncertainty, the ability of the distributional uncertainty estimate to differentiate the in-distribution set (Q) and the distractor sets (D1)–(D4) should not be affected. Moreover, also based on the mathematical theory where only data uncertainty is conditioned on distributional uncertainty and not the other way around, the distributional uncertainty score distributions of the sets should not be affected in case of low data uncertainty.

Similarly to the previous experiment, we first filter the sets before computing the distributions. This time, we use only the samples of the sets (Q), (D1)–(D4) that had a low data uncertainty score in Figure 8b. As criteria, we use a distance of 1.5 standard deviations with respect to the query set (Q) to the mean of the query set (Q), as defined in Equation (50).

$$\mathcal{H}\left(\Sigma^{(D)}\right) < \mathbb{E}\left\{\mathcal{H}\left(\Sigma_i^{(D),\mathcal{Q}}\right)\right\}_{i=1}^{Q} + 1.5 \cdot \sqrt{\mathbb{V}\mathrm{ar}\left\{\mathcal{H}\left(\Sigma_i^{(D),\mathcal{Q}}\right)\right\}_{i=1}^{Q}} \tag{50}$$

Therefore, the threshold for filtering the sets is $\tau_{\downarrow\mathcal{H}\left(\Sigma^{(D)}\right)} = 2705$. The remaining number of images in each set, along with the percentage of the original set size that remains, is provided in Table 10.

Table 10. Statistics for subsets of images with low data uncertainty ($\downarrow\mathcal{H}(\Sigma^{(D)})$).

Set		# of Images	Remaining
Query	(Q)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$	3110	92.3%
Full Human	(D1)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$	204	73.4%
Human Fragments	(D2)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$	825	47.7%
Objects	(D3)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$	110	19.4%
Background/Blur	(D4)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$	30	13.4%

In all sets, at least 30 images remain. Therefore, we can display the distributions for all sets. Figure 8e shows the distributions of the reduced sets. As expected, the distributional

uncertainty score distributions of the reduced sets (Q), (D1)–(D4)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ in Figure 8e are very similar to the distributional uncertainty score distributions of the original sets (Q), (D1)–(D4) in Figure 8c. The means of the distributions of the sets (Q), (D1)–(D4)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ remain in the correct order. The distractor set (D1)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ still largely overlaps with the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ for the previously described reasons. The distractor sets (D2)–(D4)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ with a higher OOD degree share only a small overlap with the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$. The mean of the distributions of the two distractor sets (D3)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ and (D4)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$ with the highest OOD degree move slightly closer to the in-distribution set (Q)$^{\downarrow\mathcal{H}(\Sigma^{(D)})}$. Based on Figure 4d, this can be attributed to the fact that the samples in the original sets (D3) and (D4) with the highest estimated distributional uncertainty also have an estimated data uncertainty score higher than the threshold $\tau_{\downarrow\mathcal{H}\left(\Sigma^{(D)}\right)}$.

We essentially observe what we expected from the mathematical theory. The distributional uncertainty score distributions are barely affected by filtering based on low estimated data uncertainty. The distributional uncertainty estimate is still able to predict the OOD degree. Therefore, this observation gives strong support to our hypothesis that our estimate of distributional uncertainty is a much better fit than the data uncertainty estimate for predicting the OOD degree. Furthermore, the alignment of the observation and the mathematical theory supports our hypothesis that the proposed extraction of distributional uncertainty as variance of the learned variance vector is a mathematically grounded distributional uncertainty estimate.

Data Uncertainty in Case of High Distributional Uncertainty

In case of low estimated distributional uncertainty, we saw very sharp distributions of the data uncertainty scores for all the sets. Below, we investigate how the data uncertainty estimate behaves in case of high estimated distributional uncertainty. Based on the mathematical theory, in this case, we expect the data uncertainty score distributions to be very wide since the model's ability to accurately predict the data uncertainty is expected to be reduced under the condition of high distributional uncertainty. If we verify this, it would imply that we should not trust the data uncertainty estimate in this situation.

Again, we first filter the sets before computing the distributions. This time, we use only the samples of the sets (Q), (D1)–(D4) that had a high distributional uncertainty score in Figure 8c. As criteria, we use a distance of 3 standard deviations with respect to the query set (Q) to the mean of the query set (Q), as defined in Equation (51).

$$\mathcal{H}\left(\Sigma^{(V)}\right) > \mathbb{E}\left\{\mathcal{H}\left(\Sigma_i^{(V),\mathcal{Q}}\right)\right\}_{i=1}^{Q} + 3 \cdot \sqrt{\mathbb{V}\mathrm{ar}\left\{\mathcal{H}\left(\Sigma_i^{(V),\mathcal{Q}}\right)\right\}_{i=1}^{Q}} \tag{51}$$

Therefore, the threshold for filtering the sets is $\tau_{\uparrow\mathcal{H}\left(\Sigma^{(V)}\right)} = -5794$. The remaining number of images in each set, along with the percentage of the original set size that remains, is provided in Table 11.

Table 11. Statistics for subsets of images with high distributional uncertainty ($\uparrow\mathcal{H}(\Sigma^{(V)})$).

Set		# of Images	Remaining
Query	(Q)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$	38	1.1%
Full Human	(D1)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$	26	9.4%
Human Fragments	(D2)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$	685	39.6%
Objects	(D3)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$	401	70.7%
Background/Blur	(D4)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$	188	83.9%

In all sets, at least 26 images remain. Therefore, we can display the distributions for all sets. Figure 8f shows the distributions of the reduced sets.

As expected, the distributions for the distractor sets (D1)–(D4)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$ are very wide. The order of the distractor sets (D2)–(D4)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$ with the highest OOD degree are correct. We hypothesize that this observation can be attributed to the correlation between data uncertainty and distributional uncertainty. However, for the distractor set (D1)$^{\uparrow\mathcal{H}(\Sigma^{(V)})}$ with the lowest OOD degree, we observe the highest mean value of the data uncertainty score distribution. This can be attributed to the previous observation that some samples in (D1) have poor image quality and are correctly estimated to have high data uncertainty. Simultaneously, these samples have relatively high estimated distributional uncertainty. This would suggest that the image corruptions are so severe that the images are identified as being out-of-distribution.

Essentially, our observation of wide data uncertainty score distributions in case of high distributional uncertainty matches the expectation from mathematical theory. This provides further evidence that our proposed method for estimating data uncertainty and distributional uncertainty is mathematically grounded.

Summary

From our experiments regarding the ability to predict the extent to which an input is out-of-distribution, we conclude that our *distributional uncertainty* estimate is a suitable measure of the OOD degree. Since all our observations align with mathematical theory, we have strong support for our hypothesis that the proposed extraction of the variance of the learned variance vector is a mathematically grounded estimate of distributional uncertainty. Furthermore, we observed that the model uncertainty and data uncertainty estimates are less suitable for predicting the OOD degree. This also aligns with mathematical theory and findings in related work [57].

5. Conclusions

In the context of autonomous agents, it is of critical importance to estimate the uncertainty in the prediction of deep-learning-based modules utilized for the perception of their surrounding environment. In the event that an autonomous agent interacts with humans, the deployment of uncertainty as a metric for trustworthiness is of particular significance, as misjudgments have the potential to result in substantial consequences, including damage or injury. However, if the network output is not a regression or classification, but instead a feature vector destined for subsequent processing modules, the estimation of uncertainty becomes more challenging. Furthermore, if the estimated uncertainty indicates that the prediction is unreliable, it may be beneficial to identify the underlying cause in order to enhance the prediction. Accordingly, three types of uncertainty can be extracted: model uncertainty, data uncertainty, and distributional uncertainty.

We proposed an architecture that builds upon the UAL approach [25] to estimate the three types of uncertainty. In accordance with related work [22,23,28], we utilized the person re-identification task for benchmarking purposes. In particular, we conducted our experiments on the Market-1501 dataset [68], which is the most frequently used within this context.

Our comprehensive experimental analysis demonstrated that the estimates for the three types of uncertainties exhibited the characteristics expected from mathematical theory. Furthermore, we showed that the data uncertainty and model uncertainty estimates provide valuable information for refining the feature vector, thereby enhancing the re-identification quality. The utilization of refined feature vectors by our approach, named uncertainty-based embedding refinement (UBER), results in a superior performance compared to state-of-the-art approaches that also employ uncertainty estimation for feature vectors. The improvement of our approach UBER over UAL [25] is higher than the improvement of UAL [25] over the already strong baseline BOT [66]. Additionally, we demonstrated that the distributional uncertainty estimate can reliably quantify the extent to which an input image is out-of-distribution. In contrast, the other types of uncertainties, which were taken into consideration in this context by Schwaiger et al. [57], are less reliable measures. The

capacity to predict when an image is out-of-distribution, and therefore when the prediction cannot be trusted, is an important ability for an autonomous agent.

In summary, we proposed an architecture to extract mathematically grounded estimates for the three types of uncertainties and demonstrated the individual value of these uncertainty types in our experiments.

Future Work

Our findings demonstrate that the utilization of uncertainties for refining the feature vector is advantageous for enhancing the performance of person re-identification on the Market-1501 dataset. Future work should investigate whether our approach, UBER, can also be employed to enhance the performance in related feature-vector-based tasks, such as face recognition, object re-identification, content-based image retrieval, zero-shot action recognition, or natural language processing. It should also be investigated whether it can be employed for cross-dataset person re-identification. In this context, it would be particularly interesting to examine whether other backgrounds are assessed as being out-of-distribution and whether this information can be attributed to specific feature vector components that encode context information.

Furthermore, as a methodical investigation, future work could incorporate loss functions that operate directly on the basis of distributions, such as the Bayesian Triplet Loss [36].

Our comprehensive experiments regarding the distribution of uncertainty scores for different distractor sets revealed some relations between estimated data uncertainty and estimated distributional uncertainty. In future work, qualitative analyses could examine the characteristics of images that are judged to have high data uncertainty and low distributional uncertainty, or similar combinations.

Moreover, we provided a mathematical and methodological foundation for the utilization of uncertainties in feature-vector-based tasks. It is now up to future research to demonstrate the potential of this approach when deployed in practical autonomous systems. This will reveal whether the utilization and visualization of uncertainties enhance user trust and acceptance of these autonomous systems.

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

Funding: This research was funded by the Carl Zeiss Foundation as part of the project 'Engineering for Smart Manufacturing (E4SM)—Engineering of machine learning-based assistance systems for data-intensive industrial scenarios' (funding number: P2017-01-005).

Data Availability Statement: The data and the code are available at https://www.tu-ilmenau.de/neurob/data-sets-code/uncertainties-for-embeddings (accessed on 22 August 2024).

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

References

1. Daily, M.; Medasani, S.; Behringer, R.; Trivedi, M. Self-driving cars. *Computer* **2017**, *50*, 18–23. [CrossRef]
2. Badue, C.; Guidolini, R.; Carneiro, R.V.; Azevedo, P.; Cardoso, V.B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixao, T.M.; Mutz, F.; et al. Self-driving cars: A survey. *Expert Syst. Appl.* **2021**, *165*, 113816. [CrossRef]
3. Wang, L.; Gao, R.; Váncza, J.; Krüger, J.; Wang, X.V.; Makris, S.; Chryssolouris, G. Symbiotic human-robot collaborative assembly. *CIRP Ann.* **2019**, *68*, 701–726. [CrossRef]
4. Eisenbach, M.; Aganian, D.; Köhler, M.; Stephan, B.; Schröter, C.; Gross, H.M. Visual Scene Understanding for Enabling Situation-Aware Cobots. In Proceedings of the International Conference on Automation Science and Engineering (CASE), IEEE, Lyon, France, 23–27 August 2021.
5. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning (ICML), PMLR, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.

6. Kendall, A.; Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; Volume 31, pp. 5580–5590.
7. Gawlikowski, J.; Tassi, C.R.N.; Ali, M.; Lee, J.; Humt, M.; Feng, J.; Kruspe, A.; Triebel, R.; Jung, P.; Roscher, R.; et al. A survey of uncertainty in deep neural networks. *Artif. Intell. Rev.* **2023**, *56*, 1513–1589. [CrossRef]
8. Abdar, M.; Pourpanah, F.; Hussain, S.; Rezazadegan, D.; Liu, L.; Ghavamzadeh, M.; Fieguth, P.; Cao, X.; Khosravi, A.; Acharya, U.R.; et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Inf. Fusion* **2021**, *76*, 243–297. [CrossRef]
9. Mena, J.; Pujol, O.; Vitrià, J. A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–35. [CrossRef]
10. He, W.; Jiang, Z. A survey on uncertainty quantification methods for deep neural networks: An uncertainty source perspective. *arXiv* **2023**, arXiv:2302.13425.
11. Kraus, F.; Dietmayer, K. Uncertainty estimation in one-stage object detection. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (itsc), IEEE, Auckland, New Zealand, 27–30 October 2019; pp. 53–60.
12. Harakeh, A.; Smart, M.; Waslander, S.L. Bayesod: A bayesian approach for uncertainty estimation in deep object detectors. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Paris, France, 31 May–31 August 2020; pp. 87–93.
13. Gasperini, S.; Haug, J.; Mahani, M.A.N.; Marcos-Ramiro, A.; Navab, N.; Busam, B.; Tombari, F. CertainNet: Sampling-free uncertainty estimation for object detection. *IEEE Robot. Autom. Lett.* **2021**, *7*, 698–705. [CrossRef]
14. Schubert, M.; Kahl, K.; Rottmann, M. Metadetect: Uncertainty quantification and prediction quality estimates for object detection. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, Shenzhen, China, 18–22 July 2021; pp. 1–10.
15. Jungo, A.; Reyes, M. Assessing reliability and challenges of uncertainty estimations for medical image segmentation. In Proceedings of the Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, 13–17 October 2019; Proceedings, Part II 22; Springer: Berlin/Heidelberg, Germany, 2019; pp. 48–56.
16. Mehrtash, A.; Wells, W.M.; Tempany, C.M.; Abolmaesumi, P.; Kapur, T. Confidence calibration and predictive uncertainty estimation for deep medical image segmentation. *IEEE Trans. Med. Imaging* **2020**, *39*, 3868–3878. [CrossRef]
17. Hu, P.; Sclaroff, S.; Saenko, K. Uncertainty-aware learning for zero-shot semantic segmentation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21713–21724.
18. Huang, P.Y.; Hsu, W.T.; Chiu, C.Y.; Wu, T.F.; Sun, M. Efficient uncertainty estimation for semantic segmentation in videos. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 520–535.
19. Guo, H.; Wang, H.; Ji, Q. Uncertainty-guided probabilistic transformer for complex action recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 20052–20061.
20. Vilnis, L.; McCallum, A. Word representations via gaussian embedding. *arXiv* **2014**, arXiv:1412.6623.
21. Shi, Y.; Jain, A.K.; Kalka, N.D. Probabilistic Face Embeddings. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6901–6910.
22. Taha, A.; Chen, Y.T.; Yang, X.; Misu, T.; Davis, L. Exploring uncertainty in conditional multi-modal retrieval systems. *arXiv* **2019**, arXiv:1901.07702.
23. Taha, A.; Chen, Y.T.; Misu, T.; Shrivastava, A.; Davis, L. Unsupervised data uncertainty learning in visual retrieval systems. *arXiv* **2019**, arXiv:1902.02586.
24. Hama, K.; Matsubara, T.; Uehara, K.; Cai, J. Exploring uncertainty measures for image-caption embedding-and-retrieval task. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2021**, *17*, 1–19. [CrossRef]
25. Dou, Z.; Wang, Z.; Chen, W.; Li, Y.; Wang, S. Reliability-aware prediction via uncertainty learning for person image retrieval. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 588–605.
26. Malinin, A.; Gales, M.J.F. Predictive Uncertainty Estimation via Prior Networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; Volume 31, pp. 7047–7058.
27. Nandy, J.; Hsu, W.; Lee, M.L. Towards Maximizing the Representation Gap between In-Domain and Out-of-Distribution Examples. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–12 December 2020; Volume 33, pp. 9239–9250.
28. An, S.; Jammalamadaka, N.; Chong, E. Maximum entropy information bottleneck for uncertainty-aware stochastic embedding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 3809–3818.
29. Scott, T.R.; Ridgeway, K.; Mozer, M.C. Stochastic prototype embeddings. *arXiv* **2019**, arXiv:1909.11702.
30. Neelakantan, A.; Shankar, J.; Passos, A.; McCallum, A. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv* **2015**, arXiv:1504.06654.
31. Chang, J.; Lan, Z.; Cheng, C.; Wei, Y. Data uncertainty learning in face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 5710–5719.
32. Chen, K.; Lv, Q.; Yi, T. Fast and reliable probabilistic face embeddings in the wild. *arXiv* **2021**, arXiv:2102.04075.

33. Li, S.; Xu, J.; Xu, X.; Shen, P.; Li, S.; Hooi, B. Spherical confidence learning for face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 15629–15637.
34. Zhu, Q.; Mao, Q.; Zhang, J.; Huang, X.; Zheng, W. Towards A Robust Group-level Emotion Recognition via Uncertainty-Aware Learning. *arXiv* **2023**, arXiv:2310.04306.
35. Sun, J.J.; Zhao, J.; Chen, L.C.; Schroff, F.; Adam, H.; Liu, T. View-invariant probabilistic embedding for human pose. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part V 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 53–70.
36. Warburg, F.; Jørgensen, M.; Civera, J.; Hauberg, S. Bayesian triplet loss: Uncertainty quantification in image retrieval. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 12158–12168.
37. Li, X.; Dai, Y.; Ge, Y.; Liu, J.; Shan, Y.; Duan, L.Y. Uncertainty modeling for out-of-distribution generalization. *arXiv* **2022**, arXiv:2202.03958.
38. Karpukhin, I.; Dereka, S.; Kolesnikov, S. Probabilistic embeddings revisited. *Vis. Comput.* **2023**, *40*, 4373–4386. [CrossRef]
39. Liang, S.; Dai, W.; Wei, Y. Uncertainty Learning for Noise Resistant Sketch-Based 3D Shape Retrieval. *IEEE Trans. Image Process.* **2021**, *30*, 8632–8643. [CrossRef] [PubMed]
40. Zhang, B.; Wonka, P. Point cloud instance segmentation using probabilistic embeddings. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 8883–8892.
41. Li, W.; Huang, X.; Lu, J.; Feng, J.; Zhou, J. Learning probabilistic ordinal embeddings for uncertainty-aware regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 13896–13905.
42. Zhou, J.; Tang, Y.; Su, B.; Wu, Y. Unsupervised Embedding Learning from Uncertainty Momentum Modeling. *arXiv* **2021**, arXiv:2107.08892.
43. Park, J.; Lee, J.; Kim, I.J.; Sohn, K. Probabilistic representations for video contrastive learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 14711–14721.
44. Kirchhof, M.; Kasneci, E.; Oh, S.J. Probabilistic contrastive learning recovers the correct aleatoric uncertainty of ambiguous inputs. In Proceedings of the International Conference on Machine Learning (ICML). PMLR, Seattle, WA, USA, 30 November–30 December 2023; pp. 17085–17104.
45. Upadhyay, U.; Karthik, S.; Mancini, M.; Akata, Z. Probvlm: Probabilistic adapter for frozen vison-language models. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 1899–1910.
46. Janiak, D.; Binkowski, J.; Bielak, P.; Kajdanowicz, T. Enhancing Out-of-Distribution Detection Through Stochastic Embeddings in Self-supervised Learning. In Proceedings of the International Conference on Computational Science, Málaga, Spain, 1–4 June 2024; Springer: Berlin/Heidelberg, Germany; pp. 337–351.
47. Deng, Z.; Li, D.; Song, Y.Z.; Xiang, T. Robust Target Training for Multi-Source Domain Adaptation. In Proceedings of the British Machine Vision Conference (BMVC), London, UK, 21–24 November 2022.
48. Fang, B.; Wu, W.; Liu, C.; Zhou, Y.; Song, Y.; Wang, W.; Shu, X.; Ji, X.; Wang, J. Uatvr: Uncertainty-adaptive text-video retrieval. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 13723–13733.
49. Ji, Y.; Wang, J.; Gong, Y.; Zhang, L.; Zhu, Y.; Wang, H.; Zhang, J.; Sakai, T.; Yang, Y. Map: Multimodal uncertainty-aware vision-language pre-training model. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 23262–23271.
50. Wu, X.; Li, H.; Luo, Y.; Cheng, X.; Zhuang, X.; Cao, M.; Fu, K. Uncertainty-aware sign language video retrieval with probability distribution modeling. *arXiv* **2024**, arXiv:2405.19689.
51. Yu, T.; Li, D.; Yang, Y.; Hospedales, T.M.; Xiang, T. Robust person re-identification by modelling feature uncertainty. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 552–561.
52. Zhang, Y.; Zheng, Z.; He, B.; Sun, L. Learning Posterior and Prior for Uncertainty Modeling in Person Re-Identification. *arXiv* **2020**, arXiv:abs/2007.08785.
53. Long, X.; Hu, R.; Xu, X. Variance Weight Distribution Network Based Noise Sample Learning for Robust Person Re-identification. In Proceedings of the Computer Graphics International (CGI), Shanghai, China, 28 August–2 September 2021; pp. 101–112.
54. Li, Z.; Li, Z.; Shi, Y.; Ling, H.; Chen, J.; Wang, R.; Li, P. Uncertainty-Guided Person Search Model with Auxiliary Shallow Feature Exploration. In Proceedings of the ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Seoul, Republic of Korea, 14–19 April 2024; pp. 4795–4799.
55. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
56. Chen, K.; Yi, T.; Lv, Q. Fast and reliable probabilistic face embeddings based on constrained data uncertainty estimation. *Image Vis. Comput.* **2022**, *121*, 104429. [CrossRef]
57. Schwaiger, A.; Sinhamahapatra, P.; Gansloser, J.; Roscher, K. Is uncertainty quantification in deep learning sufficient for out-of-distribution detection? In Proceedings of IJCAI-Workshop on Artificial Intelligence Safety (AISafety), Yokohama, Japan, 7–8 January 2021.

58. Müller, S.; Wengefeld, T.; Trinh, T.Q.; Aganian, D.; Eisenbach, M.; Gross, H.M. A multi-modal person perception framework for socially interactive mobile service robots. *Sensors* **2020**, *20*, 722. [CrossRef]
59. Salehi, M.; Mirzaei, H.; Hendrycks, D.; Li, Y.; Rohban, M.H.; Sabokrou, M. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv* **2021**, arXiv:2110.14051.
60. Yang, J.; Zhou, K.; Li, Y.; Liu, Z. Generalized out-of-distribution detection: A survey. *Int. J. Comput. Vis.* **2024**, 1–28. [CrossRef]
61. Jin, X.; Lan, C.; Zeng, W.; Chen, Z. Uncertainty-aware multi-shot knowledge distillation for image-based object re-identification. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11165–11172.
62. Geng, Y.; Han, Z.; Zhang, C.; Hu, Q. Uncertainty-aware multi-view representation learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 19–21 May 2021; Volume 35, pp. 7545–7553.
63. Neculai, A.; Chen, Y.; Akata, Z. Probabilistic compositional embeddings for multimodal image retrieval. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 4547–4557.
64. Kiureghian, A.D.; Ditlevsen, O. Aleatory or epistemic? Does it matter? *Struct. Saf.* **2009**, *31*, 105–112. [CrossRef]
65. Gal, Y.; Ghahramani, Z. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.
66. Luo, H.; Gu, Y.; Liao, X.; Lai, S.; Jiang, W. Bag of Tricks and a Strong Baseline for Deep Person Re-Identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019; pp. 1487–1495.
67. Hermans, A.; Beyer, L.; Leibe, B. In defense of the triplet loss for person re-identification. *arXiv* **2017**, arXiv:1703.07737.
68. Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Wang, J.; Tian, Q. Scalable Person Re-identification: A Benchmark. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1116–1124.
69. He, L.; Liao, X.; Liu, W.; Liu, X.; Cheng, P.; Mei, T. FastReID: A Pytorch Toolbox for General Instance Re-identification. *arXiv* **2020**, arXiv:abs/2006.02631.
70. 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 (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
71. Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. Random erasing data augmentation. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 13001–13008.
72. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
73. Ristani, E.; Tomasi, C. Features for multi-target multi-camera tracking and re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 6036–6046.
74. Zheng, L.; Yang, Y.; Hauptmann, A.G. Person re-identification: Past, present and future. *arXiv* **2016**, arXiv:1610.02984.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Explainable Machine Learning Model to Accurately Predict Protein-Binding Peptides

Sayed Mehedi Azim [1,†], Aravind Balasubramanyam [2,†], Sheikh Rabiul Islam [2,3], Jinglin Fu [1,3] and Iman Dehzangi [1,2,*]

[1] Center for Computational and Integrative Biology, Rutgers University, Camden, NJ 08102, USA; sa1996@camden.rutgers.edu (S.M.A.); jf604@camden.rutgers.edu (J.F.)
[2] Department of Computer Science, Rutgers University, Camden, NJ 08102, USA; ab2104@scarletmail.rutgers.edu (A.B.); sheikh.islam@rutgers.edu (S.R.I.)
[3] Department of Chemistry, Rutgers University, Camden, NJ 08102, USA
* Correspondence: i.dehzangi@rutgers.edu
† These authors contributed equally to this work.

Abstract: Enzymes play key roles in the biological functions of living organisms, which serve as catalysts to and regulate biochemical reaction pathways. Recent studies suggest that peptides are promising molecules for modulating enzyme function due to their advantages in large chemical diversity and well-established methods for library synthesis. Experimental approaches to identify protein-binding peptides are time-consuming and costly. Hence, there is a demand to develop a fast and accurate computational approach to tackle this problem. Another challenge in developing a computational approach is the lack of a large and reliable dataset. In this study, we develop a new machine learning approach called PepBind-SVM to predict protein-binding peptides. To build this model, we extract different sequential and physicochemical features from peptides and use a Support Vector Machine (SVM) as the classification technique. We train this model on the dataset that we also introduce in this study. PepBind-SVM achieves 92.1% prediction accuracy, outperforming other classifiers at predicting protein-binding peptides.

Keywords: enzyme; peptide; binding; support vector machine; physicochemical features; sequential features

1. Introduction

Enzymes are crucial for the biological functions of living organisms. They serve as catalysts to catalyze and regulate biochemical reaction pathways [1,2]. Searching ligands to modulate enzyme functions is one of the central topics for disease diagnosis and therapeutics [3–5]. Recent studies suggest that peptides are promising molecules for modulating enzyme function due to their advantages in large chemical diversity, biocompatibility, and well-established methods for library synthesis [6–9]. For example, peptides and their derivatives can be used to inhibit or activate enzyme catalysis, mediate cell receptor response, and facilitate transmembrane delivery [7,10].

Traditional ligands and drug discovery rely on the synthesis of large chemical libraries. High-throughput screening or selection is often used to identify ligands for binding to a target protein, such as in vitro mRNA display [8], phage display [11], bead-based library screening [12], protein direct evolution [13], aptamer selection [14], peptide microarray [7,10,15], and fragment-based design of small molecules [16]. Unfortunately, these approaches are all very time-consuming and costly. The recent progress of machine learning (ML) and artificial intelligence (AI) have advanced the chemical discovery process [17]. For example, by using a dataset of existing molecular libraries with known functions, ML can be trained to analyze large datasets of ligand structures, sequences, and binding affinities and to predict potential interactions more accurately [18].

During the past two decades, a wide range of ML models has been proposed to predict different types of protein-binding peptides [19–23]. These studies can generally be categorized into two groups. First are studies that introduced new features to extract more discriminatory information for the prediction task [24–26]. These studies mainly extract features based on sequential, structural, physicochemical, and evolutionary-based information [27,28].

For instance, Yan and Zhou used chemical interactions and physicochemical properties of amino acids for feature extraction [29]. Later on, Romero-Molina et al. extended the use of physicochemical properties and also included sequential-based features to predict protein-binding peptides [24]. In a different study, Wei et al., employed structural features to detect therapeutic peptides [27]. The structural features are mostly extracted based on the predicted values of the protein local or secondary structure of the proteins using tools such as PSIPRED or SPIDER [30–32]. Other studies, such as Akbar et al., used evolutionary information to add to former types of features for better performance [33].

Recent studies have shifted their focus to using Deep Learning (DL) models to automatically extract features from protein sequences instead of different types of properties [33–36]. However, extracting effective features is still considered a common and effective method to tackle this problem [37].

The second group of studies aims to use effective ML and DL models to predict protein-binding peptides. Among traditional machine learning models, Support Vector Machine (SVM), Random Forest (RF), and Adaptive Boosting have been shown to be very effective for this task [26,37–42]. As mentioned earlier, recent studies have shifted their focus to using DL to tackle this problem. However, for smaller datasets, complex DL models cannot be trained properly. Instead, traditional ML methods demonstrate better performance for these cases. In addition, studying the explainability of the complex DL models is more challenging. Therefore, there is a demand for an accurate and explainable model to tackle this problem. Another challenge is to have reliable and large enough datasets to use as a benchmark to train machine learning models to predict common protein-binding peptides. Most of the currently used datasets are either very small or have low resolution.

This study addresses these two challenges by generating a reliable and relatively large peptide-binding protein dataset and proposing an explainable machine learning model. Here, we develop PepBind-SVM to predict protein-binding peptides. To build this model, we extract different sequential and physicochemical features from peptides and use an SVM as the classification technique. We train this model on the dataset that we also introduce in this study. PepBind-SVM outperforms other classifiers in predicting protein-binding peptides. PepBind-SVM, its source code, and our newly generated protein binding peptides are publicly available at https://github.com/MLBC-lab/pepbind-SVM (accessed on 1 September 2024).

2. Dataset Preparation

To generate our own dataset, we first prepared the peptides using an experimental approach. We then filtered the data to generate a training and independent test set to build and validate our machine learning models.

2.1. Microarray Fabrication

Here, we use 10,000 20-mer peptides produced at Arizona State University [7,10,43]. These distinct polypeptide sequences were spotted in duplicate onto a glass slide possessing an amino-silane surface coating. Each polypeptide is 20 residues long, with 17 positions randomly chosen from 19 amino acids (excluding cysteine). The C-terminal three positions of each peptide was a glycine–serine–cysteine (GSC) linker, which was used for conjugating peptides to amino-silane surfaces via a maleimide linker, sulfosuccinimidyl 4-(N-maleimidomethyl) cyclohexane-1-carboxylate (Sulfo-SMCC; Pierce, Rockford, IL, USA). The array printing was done using a Telechem Nanoprint60 (Currently Arrayit, Sunnyvale, CA, USA) hot spotted approximately 500 pL of 1 mg/mL peptide per feature on glass

slides with 48 Telechem series SMP2-style 946 titanium pins (Currently Arrayit, Sunnyvale, CA, USA).

2.2. Microarray Binding

We labeled β-Gal with Alexa Fluor 647 (Invitrogen) and diluted it to 5 nM in bovine serum albumin (BSA) buffer (1 × PBS with 3% (v/v) BSA and 0.05% Tween 20). The activity of the labeled β-Gal was ~75% that of the wild type in solution. Binding of enzymes on the microarray was performed as described in [43,44]. Briefly, a microarray was first pre-washed with surface cleaning solvent (7.33% (v/v) acetonitrile, 37% isopropyl alcohol, and 0.55% trifluoroacetic acid in water), followed by the treatment of a capping buffer (3% (v/v), bovine serum albumin (BSA), 0.02% (v/v) mercaptohexanol, and 0.05% (v/v) Tween 20 in 1 × PBS) to block any active SMCC linker on the array surface. Then, the array was incubated with a solution containing 5 nM Alexa 647-labeled β-Gal or PEP-1–β-Gal complex for two hours, allowing the enzyme to bind with peptides on the array surface. After washing off unbound enzymes, the array was read by a standard array reader (PerkinElmer, Waltham, MA, USA) with color scanning using 647 nm laser lines. The array was then read by a standard array reader (PerkinElmer, Waltham, MA, USA) with dual color scanning using 488 and 647 nm laser lines.

2.3. Generating Training and Testing Data for the Machine Learning Model

After generating the peptides, we prepared training and testing datasets to build our machine learning model in the following manner. We first calculated the affinity of the peptides to interact with proteins and scored each peptide with a number between 0 and 1. After that, we defined a conservative threshold to select protein-binding and non-binding peptides. In this way, we made sure that the protein-binding peptides in our dataset are actual binding and positive samples, while non-binding peptides are actual negative samples. In this way, the model can effectively learn the patterns. Based on the corresponding scores, the number of protein-binding peptides was much smaller than that of non-binding peptides.

The number of non-binding peptides was four times more than the number of binding peptides. We considered this ratio for building our training and testing datasets.

We selected those peptides with over 0.8 (out of 1) affinity of binding to proteins as positive and those peptides with less than 0.2 (out of 1) affinity of binding to proteins as negative samples. In this way, our dataset could be more robust and have clear-cut positive and negative samples. Peptides with binding affinity between 0.2 and 0.8 cannot be accurately categorized as binding or non-binding. Therefore, to make sure that our true positive and true negative samples were accurate, we discarded them. The aim of this study is to accurately identify protein-binding and non-binding peptides. These thresholds were selected experimentally and by studying the similarity of samples with each other. As a result, we generated a training dataset containing 1600 samples. This set was balanced to contain an equal representation of classes, with 800 samples labeled as positives and 800 as negatives. Such a balanced dataset aids in preventing model bias toward the more dominant class. However, to make sure that our model performs well on the actual data, we considered the actual imbalance when we built the testing dataset. For the independent testing, a dataset comprising 1000 samples was prepared. This dataset includes 200 positive samples and 800 negative samples, intentionally skewed to reflect a real-world scenario where negative instances are more prevalent than positive ones.

3. Feature Generation

To encapsulate the complexity and intrinsic characteristics of peptides, we engineered features based on n-gram models, specifically monograms and bigrams, and incorporated the peptides' physicochemical properties. Initially, several n-lengths were computed recursively. However, no improvement was shown for subsets of length greater than 2.

Therefore, this study is restricted to monograms and bigrams as a feature set. The feature generation process is explained as follows.

3.1. Composition and Occurrence (Monogram)

To generate the occurrence, we count the number of each amino acid along the peptide. It consists of 20 features. To generate the composition, we calculate the frequency of the amino acid occurrence. To do this, we simply divide the numbers in the occurrence feature vector by the total number of amino acids in the peptide. It is shown that the performance of occurrence and composition can be different with respect to the other features being used and with respect to the problem at hand [45].

3.2. Bigram

In addition to monogram features, we explored the sequential relationship between amino acids by considering bigram, which is defined as the number of pairs of adjacent amino acids within the sequences. This approach allows the model to capture the local sequence context, which can be a determinant of the peptide's binding capability. Since there are 20 amino acids, the total number of combinations of pairs is 20×20. Hence, the Bigram feature group consists of 400 features. It has been widely used for different protein and peptide-related problems and obtained promising results [34].

3.3. Physicochemical Properties

Here, we incorporate the normalized density of 55 physicochemical properties, which is calculated as follows. For a given amino acid along the peptide sequence, we replace it with the corresponding value of its property. We then sum them all together and divide the summation by the length of the peptide, as was done in [46]. These properties include, but are not limited to, hydrophobicity, charge, and molecular weight. Normalization of these features ensures that each property contributes equally to the model's learning process, preventing any single feature from dominating due to scale differences. These features were introduced in [46] as the most effective set of physicochemical-based features.

4. Classification Method

In this study, we tried using different classifiers, and, among them, SVM demonstrated the best performance and is used to build PepBind-SVM. The result of this comparison is comprehensively presented in the Results and Discussion Section. SVM has been shown as an effective binary-class classification technique that performs well in similar studies and specifically for protein-binding peptide prediction in the literature [36–38]. SVM aims to classify the samples by identifying the maximal marginal hyperplane (MMH) to separate the data [47]. In this study, the Linear Kernel is used for our SVM model. As shown in the Result and Discussion Section, using the linear kernel, we obtained the best results compared to other kernels.

5. Evaluation Methods and Metrics

To evaluate the performance of our model, we used k-fold cross-validation on the training data and an independent testing set. In k-fold cross-validation, the training data are divided into k groups. At every step, K-1 of the groups are used for training purposes while the remaining group is being used for testing purposes. This process is repeated k times until all groups are being used as the testing set. In this way, we can effectively use our data for validation. In this study, we use k = 5. To make sure that our model is general, we also use an independent test set that is not used for training purposes.

For the evaluation of our model's performance, various metrics, including accuracy (Acc), Sensitivity (Se), Specificity (Sp), the area under the receiver–operating characteristic curve (AUC), Matthews correlation coefficient (MCC), and F1-score are used. These metrics

serve as reliable measures to assess the effectiveness and robustness of the model. The calculations for each metric are defined as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN,}$$

$$Sn = \frac{TP}{TP + FN,}$$

$$Sp = \frac{TN}{TN + FP,}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN),}}$$

$$F1\text{-}score = 2 \times \frac{Pre \times SN}{Pre + SN,}$$

where TP represents the count of true positives, TN represents the count of true negatives, FP represents the count of false positives, and FN represents the count of false negatives. These five evaluation metrics were used for results achieved for both testing dataset and five-fold cross-validation.

6. Results and Discussion

In this section, we first present our achieved results and continue with a comprehensive feature analysis. After that, we discuss the explainability of our model using Shapley Additive exPlanations (SHAP).

6.1. Selecting a Base Classifier to Build Our Model

To build our model, we use different classifiers for our extracted features. We studied several classifiers and, among them, chose RF, SVM, and gradient-boosting decision trees (XGBoost). As discussed in the Introduction Section, these three classifiers have been shown to be effective methods in predicting protein-binding peptides [26,37–42]. Introduced by Breiman, Random Forest is considered one of the best ensemble classifiers [48]. Based on the Bagging approach, RF promotes randomness to obtain better results. For RF, the employed data are first randomly divided into several groups. Then, for each group, a base decision tree is used for training using a random subset of a total number of features. Therefore, both data and features are randomly used to train each decision tree. After that, the results of all the base learners are combined to present the final classification. On the other hand, XGBoost is based on a boosting method [49]. In this model, a base classifier is trained iteratively on a subset of the data, and based on its performance, the weights for the classified samples are adjusted for the next iteration. In the next iteration, the weights of those samples that are incorrectly classified are increased to make it more costly to misclassify them. This process is being repeated several times. In the end, all those predictors in different iterations are combined to provide the final predictor. For the XGBoost, the employed base learner is gradient descent as its base learner.

We use these three classifiers with different parameters for our three extracted feature groups. We use SVM with linear and Radial Basis function kernels. We also use RF with 50, 100, and 200 decision trees. In this way, we can choose the parameters that perform the best. We used grid search as well as several values manually (those that performed well for similar studies) to tune our classifiers and optimize them. After careful comparison, we used the parameters that are general, and also their results were aligned with the outcome of the grid search to avoid overfitting. The results achieved using these classifiers for Composition, Occurrence, Bigram, and Physicochemical-based feature groups for the independent test set are presented in Table 1, Table 2, Table 3, and Table 4, respectively.

Table 1. Performance of our employed classifiers using composition feature group on the independent dataset.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	90.6	86.0	91.8	0.730	**0.945**	0.785
SVM-Linear	**92.1**	**86.0**	**93.6**	**0.765**	**0.937**	**0.813**
RF-50	89.4	84.0	90.8	0.698	0.933	0.760
RF-100	90.8	88.0	91.5	0.740	0.941	0.793
RF-200	90.5	**86.5**	91.5	0.729	0.944	0.784
XGBoost	90.1	85.5	91.3	0.718	0.942	0.775

Table 2. Performance of our employed classifiers using occurrence feature group on the independent dataset.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	**90.6**	**86.0**	**91.8**	**0.730**	**0.945**	**0.785**
SVM-Linear	90.4	85.5	91.6	0.724	0.936	0.781
RF-50	90.1	86.5	91.0	0.721	0.944	0.778
RF-100	90.6	86.5	91.6	0.732	0.943	0.786
RF-200	90.4	86.5	91.4	0.727	0.945	0.783
XGBoost	90.1	85.5	91.3	0.718	0.942	0.776

Table 3. Performance of our employed classifiers using the Physicochemical feature group on the independent dataset.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	**91.0**	**85.5**	**92.4**	**0.738**	**0.942**	**0.791**
SVM-Linear	90.4	85.5	91.6	0.724	0.936	0.781
RF-50	84.3	76.5	86.3	0.570	0.903	0.661
RF-100	84.9	78.5	86.5	0.589	0.904	0.675
RF-200	84.4	79.5	85.6	0.584	0.906	0.671
XGBoost	86.5	82.5	87.5	0.634	0.923	0.710

Table 4. Performance of our employed classifiers using the Bigram feature group on the independent dataset.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	87.7	78.5	90.0	0.644	0.928	0.719
SVM-Linear	89.6	57.5	97.6	0.647	0.918	0.689
RF-50	78.2	84.0	76.8	0.508	0.880	0.607
RF-100	79.0	85.5	77.4	0.527	0.891	0.620
RF-200	77.0	83.0	75.5	0.487	0.893	0.591
XGBoost	83.3	77.5	84.8	0.556	0.876	0.650

As shown in Tables 1–4, using SVM, we achieve the best results. After SVM, XGBoost achieves better results than RF. Among different numbers of base learners, using 100, RF achieves its best results. Still, it performs poorer than SVM and XGBoost. As shown in Table 1, using SVM with a linear kernel, we achieved 92.1% accuracy, 86.0% sensitivity, 93.6% specificity, 0.765 MCC, 0.937 AUC, and 0.813 F1 score, which outperforms other combinations. It highlights the importance of the monogram feature group. Also, better results using linear kernel demonstrate that linear transformation of the input data to a

higher dimension is better for maintaining the patterns in the data. In other words, the underlying pattern in the data can be expressed in a linear manner.

As shown in Table 4, Bigram's results are lower than those of other feature groups. This demonstrates that, with respect to the number of generated features (400), Bigram does not provide significant discriminatory information compared to other feature groups. Comparing the results in Tables 1 and 2 shows that we can achieve slightly better results by using composition. As discussed in [45] and considering the similar nature of composition and occurrence feature groups, we just use the composition feature group for the next step, which is the combination of all feature groups.

6.2. Selecting the Input Feature Groups to Build Our Model

Next, we combine composition, physicochemical, and Bigram feature groups and use our classifiers on the independent test set. The result of this experiment is shown in Table 5.

Table 5. Performance of our employed classifiers using a combination of the composition, physicochemical, and Bigram feature groups on the independent dataset.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	82.5	67.5	86.2	0.499	0.853	0.606
SVM-Linear	90.4	86.0	91.5	0.725	0.936	0.781
RF-50	85.0	78.0	86.7	0.589	0.907	0.675
RF-100	84.9	79.0	86.3	0.591	0.908	0.676
RF-200	86.3	82.0	87.3	0.628	0.922	0.705
XGBoost	89.2	84.5	90.3	0.695	0.942	0.757

As shown in Table 5, using a combination of different feature groups, we achieve lower results for all the metrics. It shows that we do not add extra discriminatory information, considering the number of additional features when combining these feature groups. Considering the number of added features, Bigram might have the highest weight in reducing the performance. Therefore, in the next step, we just use the combination of composition and physicochemical feature groups. The result of this experiment is shown in Table 6.

Table 6. Performance of our employed classifiers using a combination of the composition and physicochemical feature groups on the independent dataset.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	88.8	80.0	91.0	0.673	0.918	0.741
SVM-Linear	**90.2**	**85.5**	**91.4**	**0.720**	**0.936**	**0.777**
RF-50	87.4	80.5	89.1	0.645	0.924	0.719
RF-100	87.7	83.0	88.9	0.660	0.931	0.729
RF-200	88.2	84.0	89.3	0.673	0.935	0.740
XGBoost	89.6	86.0	90.5	0.708	0.943	0.768

As shown in Table 6, adding a physicochemical-based feature group reduces the performance compared to using composition or occurrence alone. It again highlights that the discriminatory information presented in the physicochemical-based feature group is already captured in the occurrence and composition feature group. It is actually in line with the fact that the necessary information for understanding the protein sequence is already embedded in its amino acid sequence [50]. To validate and investigate the generality and robustness of achieved results, we use our employed classifiers on the composition and the combination of all three feature groups (composition, Bigram, and physicochemical-

based feature groups) using five-fold cross-validation. The results of these experiments are presented in Tables 7 and 8, respectively.

Table 7. Performance of our employed classifiers on the composition feature group using five-fold cross-validation.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	88.4	84.5	92.4	0.760	0.925	0.871
SVM-Linear	89.1	85.4	92.9	0.784	0.931	0.887
RF-50	87.4	85.0	89.8	0.750	0.932	0.871
RF-100	88.2	86.1	90.4	0.765	0.937	0.880
RF-200	87.6	85.7	89.6	0.754	0.938	0.874
XGBoost	87.8	86.1	89.6	0.758	0.937	0.876

Table 8. Performance of our employed classifiers using a combination of the composition and physicochemical feature groups using five-fold cross-validation.

Model	Acc (%)	Sn (%)	Sp (%)	MCC	AUC	F1-Score
SVM-RBF	77.7	68.8	86.6	0.564	0.859	0.756
SVM-Linear	88.4	85.3	91.5	0.769	0.930	0.880
RF-50	83.2	80.6	85.7	0.665	0.903	0.827
RF-100	83.8	81.1	86.5	0.678	0.910	0.833
RF-200	84.0	81.6	86.5	0.682	0.910	0.836
XGBoost	87.4	86.3	88.6	0.749	0.934	0.873

As shown in Tables 7 and 8, although the results are lower than those reported on the independent test set, the trend is still similar. As shown in these tables, using the SVM classifier and composition as the input feature group, we achieve better performance than using the combination of all the employed feature groups. This highlights the generality of our achieved results. The lower results achieved on the five-fold cross-validation can be associated with using far fewer samples to test our model. Using five-fold cross-validation, we use just 80% of our data to train the model, compared to using 100% of the data when we use an independent test set. As a result, we use an SVM classifier with a linear kernel and composition feature group to build PepBind-SVM. PepBind-SVM, its source code, and our newly generated protein binding peptides are publicly available at https://github.com/MLBC-lab/pepbind-SVM (accessed on 1 September 2024).

Achieving the best results using SVM with linear kernel as the classifier and composition as feature group, which are relatively simple combinations, can be directly related to the employed dataset. We prepared the dataset with several control cases to make sure that it is a very clear representation of protein-binding and non-binding peptides. Achieving promising results using this combination demonstrates that if we use proper data with high resolution and clarity, finding the pattern in the data is much more straightforward. It is important to highlight that despite promising results, we still have a long way to go to solve this problem. Hence, for our future study, we aim at using structural and evolutionary based features to tackle this problem. We also aim to use a complex deep-learning model to solve this problem. One of the limitations of this study is regarding data cleaning. We cleaned up our data using a binding affinity of less than 0.2 for non-binding peptides and over 0.8 for binding peptides. This threshold enables us to make sure that our positive samples and negative samples are correct. In the future, we aim to collect more data and experimentally validate the proper threshold or try other thresholds with more extensive data.

Another limitation of this study is that our model is designed to identify protein-binding and non-binding peptides. However, it is not able to identify peptides with

binding affinity between 0.2 and 0.8 (uncertain proteins). Our future direction is to identify a proper threshold, which would be validated experimentally with the help of experts, for uncertain peptides and identify them.

Finally, since the employed dataset is new and was generated by us, there is no previous work with which to compare our model. However, we share this dataset along with this article for future studies to evaluate their model. Our future direction is also to use complex yet explainable models to tackle this problem. Next, we investigate the explainability of the proposed model using SHAP.

6.3. SHAP for Interpreting Model Feature Relation and Black Box Model Explanation

Shapley Additive exPlanations (SHAP) is a widely used method for interpreting machine learning models [51]. Studies have used this method to elucidate the decision-making processes of machine learning models across various types of biological data [52,53]. This technique is created based on the shapely values from cooperative game theory. Each feature in the data plays the role of the cooperative player. SHAP explains the output of machine learning models by assigning each feature an importance value for a specific prediction. Note that we considered top XAI methods such as SHAP, LIME [54], and ELI5 [55], and ultimately chose SHAP since it has been successfully used for similar problems and obtained promising results [56,57]. Shapely values present a fair method for allocating rewards among participants involved in a collaborative game. In such a game, a group of N players collaborates to generate a total payout of v. Shapely value ϕ_i assigns a fair payout to each player based on their contribution. A feature with a higher magnitude of SHAP value indicates a significant contribution of that feature to the prediction. The Shapley value ϕ_i is calculated using the following equation:

$$\phi_i = \sum_{s \subseteq \mathrm{F}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f\left(x_{s \cup \{i\}}\right) - f(x_s) \right]$$

where F is the set of all features, and S is a subset of features that does not contain feature i. As shown in this equation, ϕ_i is computed by considering all possible feature subsets S and evaluating how the prediction changes by including or excluding features i. In this study, we investigated both local and global explanations using SHAP. Local explanations are focused on providing a deeper understanding of individual predictions, while global explanations provide insight into the overall behavior of the model [58]. Our goal is to gain a comprehensive understanding of how each feature contributes to the model's decision-making in peptide-binding site prediction.

We begin by examining local explanations through the computation of SHAP values for our test dataset. We generate a waterfall plot based on all n-gram occurrence and n-gram composition features for n = 1 and n = 2. A waterfall plot is used for local explanation utilizing the SHAP values calculated for a single instance. Figure 1 illustrates waterfall plots for four distinct instances randomly selected from the test dataset.

The upper two subfigures (Figure 1a,b) depict instances for which the model shows strong confidence in assigning them to the positive class. The lower two subfigures (Figure 1c,d) show two instances for which the model exhibits strong confidence in assigning them to a negative class. Additionally, the plot provides compelling evidence in favor of utilizing only occurrence-based features in constructing tools for predicting peptide binding sites since among 840 n-gram features, the top 20 contributing features are all generated by calculating the occurrence/composition of amino acids in the sequence. This highlights that n-gram occurrence/composition features are able to provide distinguishable characteristics for peptide-binding site prediction. In addition, amino acids with electrically charged side chains, such as Arginine (R), Lysine (L), Aspartic Acid (D), and Glutamic Acid (E), constantly demonstrate higher values. It shows that charged amino acids are more likely to have an important impact on the binding or non-binding properties of the peptides.

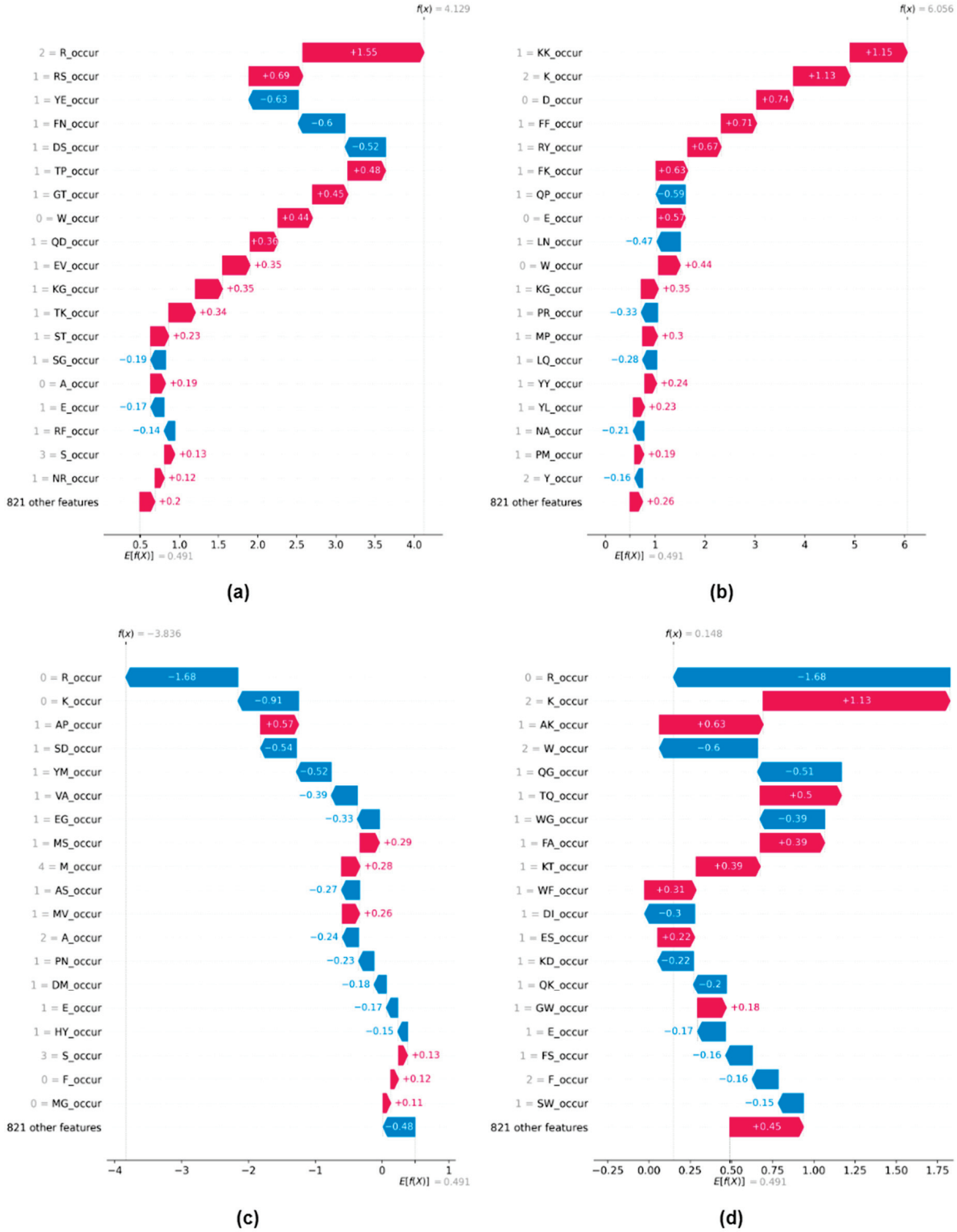


Figure 1. Local explanations using waterfall plots for n-gram occurrence and n-gram composition features for the values of n = 1 and n = 2. A prediction f(x), for instance, x, and its deviation from E[f(x)] could be measured as the sum of the feature importance values (SHAP values) of individual features. (**a**) Instance with positive prediction (p = 0.92) (**b**) Instance with positive prediction (p = 0.97) (**c**) Instance with negative prediction (p = 0.08) (**d**) Instance with negative prediction (p = 0.5).

To further investigate the efficacy of amino acid occurrence-based features in our task, we delved into global-level explanations. Leveraging the SHAP library, we computed the global feature importance, which provides a holistic view of each feature's contribution to the model's predictions across the entire dataset. We computed the global feature importance by taking the mean absolute SHAP values across all samples. Figure 2 depicts the top 20 features contributing towards model prediction across all samples. In addition, we generated a summary plot using the violine plot type from the SHAP library, which displays the distribution and density of SHAP values for their respective feature. In Figure 3, we present a violine summary plot highlighting the top feature filtered from the pool of 840 n-gram features. Together, these figures strongly indicate the pivotal role of amino acid occurrence-based features in driving the model's prediction.

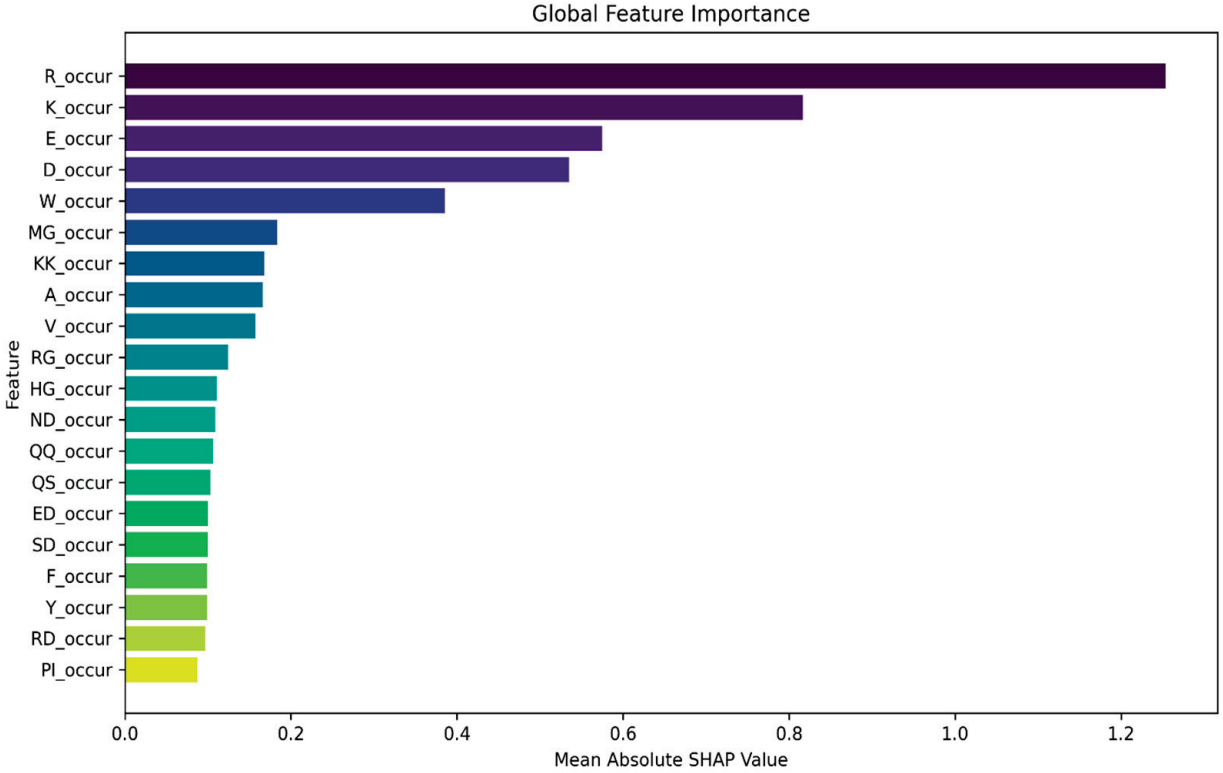


Figure 2. Global feature importance for n-gram occurrence and n-gram composition features is computed by taking the mean absolute SHAP values across all samples. This plot provides insights into the overall contribution of individual features to the model's prediction across all samples in the dataset. Only the top 20 contributing features are shown.

Finally, we explore both the local and global explanations for Monogram occurrence features. We observed that using monogram features alone for training our machine learning model yields the best performance in distinguishing protein-binding peptides from non-binding peptides. Figure 4 shows the waterfall plot for four instances from the test dataset. Figure 4a,b depict two samples for which the model predicts positive sites, and Figure 4c,d show samples for which the model is showing strong confidence in assigning them to the negative class.

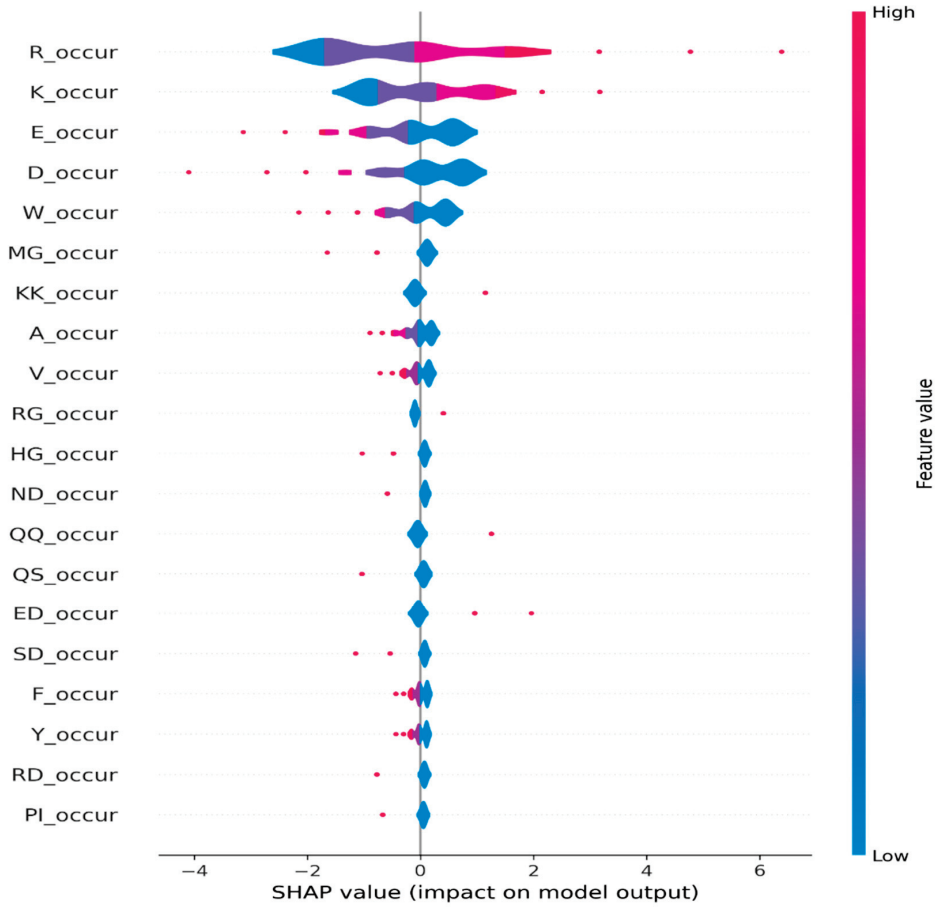


Figure 3. Global explanations using summary plots from SHAP for n-gram occurrence and n-gram composition features.

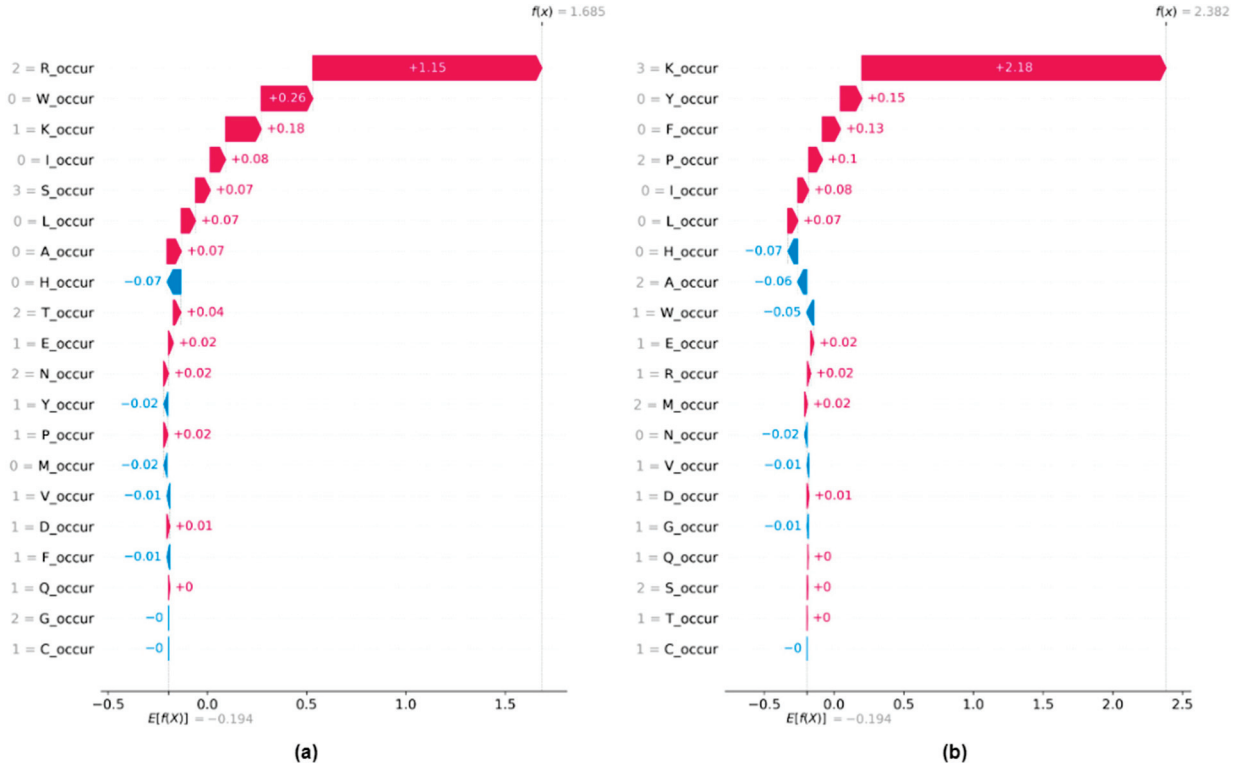


Figure 4. *Cont.*

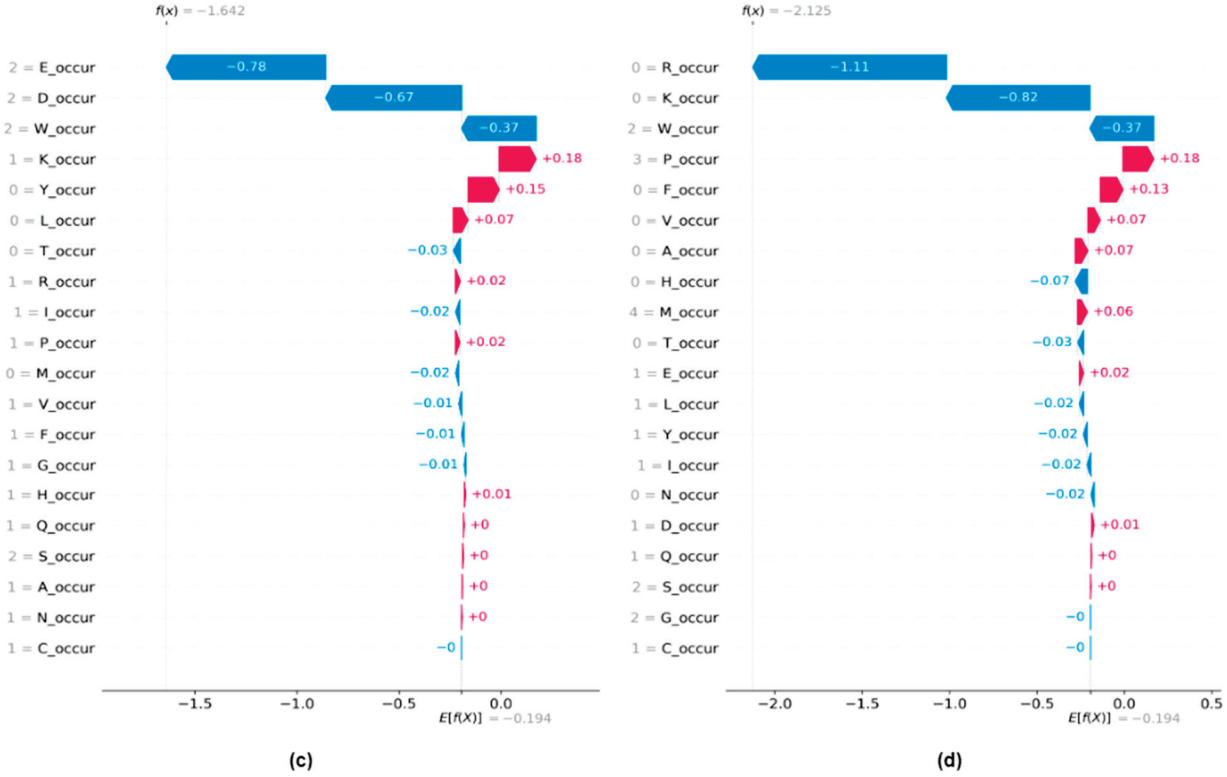


Figure 4. Local explanations using waterfall plots for monogram occurrence features. A prediction f(x), for instance, x, and its deviation from E[f(x)] could be measured as the sum of the feature importance values (SHAP values) of individual features. (**a**) Instance with positive prediction ($p = 0.91$) (**b**) Instance with positive prediction ($p = 0.96$) (**c**) Instance with negative prediction ($p = 0.11$) (**d**) Instance with negative prediction ($p = 0.06$).

Moreover, we generate a global feature importance plot using SHAP values, as depicted in Figure 5. Additionally, we generate a violine summary plot (Figure 6) for monogram features, which provides a compact representation of the distribution and variability of SHAP values for each feature.

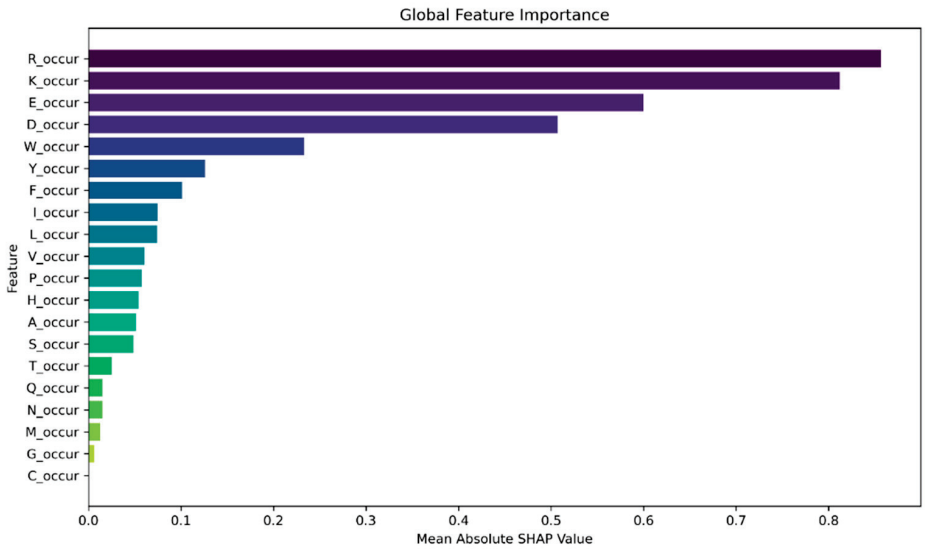


Figure 5. Global feature importance for monogram occurrence features is computed by taking the mean absolute SHAP values across all samples. Only the top 20 contributing features are shown.

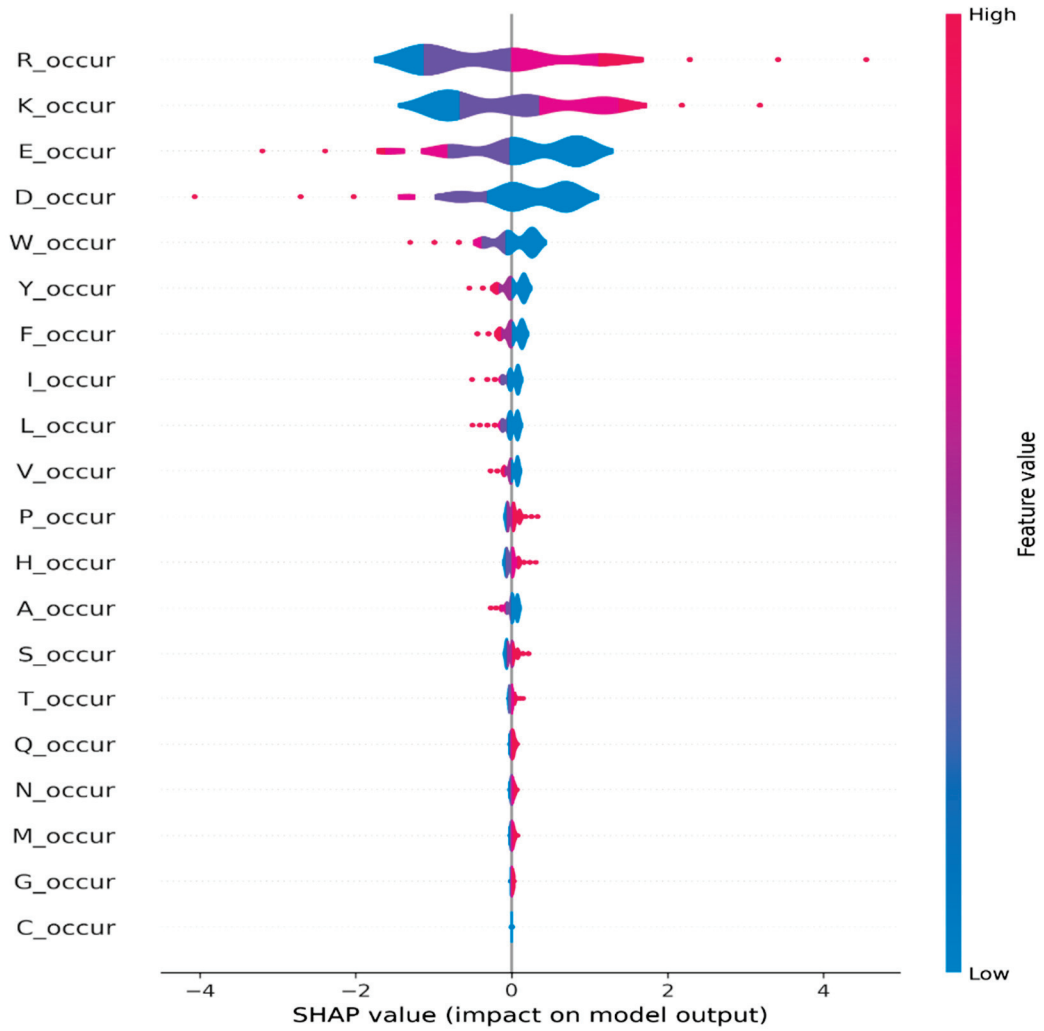


Figure 6. Global explanations using summary plots from SHAP for monogram occurrence features.

7. Conclusions

Peptides are considered important macromolecules for modulating enzyme function due to their advantages in large chemical diversity, biocompatibility, and well-established methods for library synthesis. Experimental approaches to identify functioning peptides, such as those that are protein binding, are time-consuming and costly. Hence, there is a demand to build a fast and accurate computational approach to tackle this problem. In this study, we proposed PepBind-SVM, a new machine learning-based model to accurately predict protein-binding peptides. To build such a model and make sure that it is able to identify binding patterns in peptides, we generate a new experimentally validated dataset that is released alongside this article.

We then investigate different feature sets and classifiers to identify the most effective combination. Finally, we comprehensively investigate the explainability of our model using SHAP. Our analysis demonstrates the importance of amino acids with electrically charged side chains playing an important role in protein-binding or non-binding properties of the peptides. As mentioned earlier, here we focused on proposing an accurate, explainable machine learning model to tackle this problem. Therefore, our proposed method is relatively simple. However, despite the simplicity and due to the generation of a well-prepared dataset, PepBind-SVM obtained promising results. PepBind-SVM achieves 92.1% prediction accuracy, outperforming other classifiers in predicting protein-binding peptides. For our future study, we aim to investigate other sources of features, such as structural and evolutionary attributes. We also aim to use more sophisticated deep-learning

models. PepBind-SVM, its source code, and our newly generated protein-binding peptides are publicly available at https://github.com/MLBC-lab/pepbind-SVM (accessed on 1 September 2024).

Author Contributions: Conceptualization, J.F., I.D. and S.R.I.; methodology, S.M.A., A.B., J.F. and I.D.; software, S.M.A.; validation, S.M.A., I.D., A.B. and S.R.I.; formal analysis, A.B. and S.M.A.; investigation, J.F., I.D., A.B. and S.M.A.; resources, J.F. and I.D.; data curation, J.F.; writing—original draft preparation, I.D., S.M.A., A.B. and S.R.I.; writing—review and editing, all authors. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: PepBind-SVM, its source code, and our newly generated protein-binding peptides are publicly available at https://github.com/MLBC-lab/pepbind-SVM (accessed on 1 September 2024).

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

References

1. Robinson, P.K. Enzymes: Principles and biotechnological applications. *Essays Biochem.* **2015**, *59*, 1. [CrossRef]
2. Medina-Cleghorn, D.; Nomura, D.K. Exploring metabolic pathways and regulation through functional chemoproteomic and metabolomic platforms. *Chem. Biol.* **2014**, *21*, 1171–1184. [CrossRef]
3. Drews, J. Drug discovery: A historical perspective. *Science* **2000**, *287*, 1960–1964. [CrossRef]
4. Pathan, S.U.; Kharwar, A.; Ibrahim, M.A.; Singh, S.B.; Bajaj, P. Enzymes as indispensable markers in disease diagnosis. *Bioanalysis* **2024**, *16*, 485–497. [CrossRef]
5. Vegas, A.J.; Fuller, J.H.; Koehler, A.N. Small-molecule microarrays as tools in ligand discovery. *Chem. Soc. Rev.* **2008**, *37*, 1385–1394. [CrossRef]
6. Keppler, A.; Kindermann, M.; Gendreizig, S.; Pick, H.; Vogel, H.; Johnsson, K. Labeling of fusion proteins of O6-alkylguanine-DNA alkyltransferase with small molecules in vivo and in vitro. *Methods* **2004**, *32*, 437–444. [CrossRef]
7. Fu, J.; Cai, K.; Johnston, S.A.; Woodbury, N.W. Exploring peptide space for enzyme modulators. *J. Am. Chem. Soc.* **2010**, *132*, 6419–6424. [CrossRef]
8. Roberts, R.W.; Szostak, J.W. RNA-peptide fusions for the in vitro selection of peptides and proteins. *Proc. Natl. Acad. Sci. USA* **1997**, *94*, 12297–12302. [CrossRef]
9. Greving, M.P.; Belcher, P.E.; Diehnelt, C.W.; Gonzalez-Moa, M.J.; Emery, J.; Fu, J.; Johnston, S.A.; Woodbury, N.W. Thermodynamic additivity of sequence variations: An algorithm for creating high affinity peptides without large libraries or structural information. *PLoS ONE* **2010**, *5*, e15432. [CrossRef]
10. Fu, J. Microarray Selection of Cooperative Peptides for Modulating Enzyme Activities. *Microarrays* **2017**, *6*, 8. [CrossRef]
11. Smith, G.P. Filamentous fusion phage: Novel expression vectors that display cloned antigens on the virion surface. *Science* **1985**, *228*, 1315–1317. [CrossRef] [PubMed]
12. Liu, T.; Qian, Z.; Xiao, Q.; Pei, D. High-throughput screening of one-bead-one-compound libraries: Identification of cyclic peptidyl inhibitors against calcineurin/NFAT interaction. *ACS Comb. Sci.* **2011**, *13*, 537–546. [CrossRef]
13. Guntas, G.; Mansell, T.J.; Kim, J.R.; Ostermeier, M. Directed evolution of protein switches and their application to the creation of ligand-binding proteins. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 11224–11229. [CrossRef]
14. Blind, M.; Blank, M. Aptamer selection technology and recent advances. *Mol. Ther. Nucleic Acids* **2015**, *4*, e223. [CrossRef]
15. Legutki, J.B.; Zhao, Z.G.; Greving, M.; Woodbury, N.; Johnston, S.A.; Stafford, P. Scalable high-density peptide arrays for comprehensive health monitoring. *Nat. Commun.* **2014**, *5*, 4785. [CrossRef] [PubMed]
16. Murray, C.W.; Rees, D.C. The rise of fragment-based drug discovery. *Nat. Chem.* **2009**, *1*, 187–192. [CrossRef]
17. Dybowski, R. Interpretable machine learning as a tool for scientific discovery in chemistry. *New J. Chem.* **2020**, *44*, 20914–20920. [CrossRef]
18. Goulard Coderc de Lacam, E.; Roux, B.; Chipot, C. Classifying Protein–Protein Binding Affinity with Free-Energy Calculations and Machine Learning Approaches. *J. Chem. Inf. Model.* **2024**, *64*, 1081–1091. [CrossRef]
19. Kozlovskii, I.; Popov, P. Protein–peptide binding site detection using 3D convolutional neural networks. *J. Chem. Inf. Model.* **2021**, *61*, 3814–3823. [CrossRef]
20. Ferdous, S.M.; Mugdha, S.B.S.; Dehzangi, I. New Multi-View Feature Learning Method for Accurate Antifungal Peptide Detection. *Algorithms* **2024**, *17*, 247. [CrossRef]
21. Ahmed, S.; Muhammod, R.; Khan, Z.H.; Adilina, S.; Sharma, A.; Shatabda, S.; Dehzangi, A. ACP-MHCNN: An accurate multi-headed deep-convolutional neural network to predict anticancer peptides. *Sci. Rep.* **2021**, *11*, 23676. [CrossRef] [PubMed]
22. Yan, K.; Lv, H.; Guo, Y.; Peng, W.; Liu, B. sAMPpred-GAT: Prediction of antimicrobial peptide by graph attention network and predicted peptide structure. *Bioinformatics* **2023**, *39*, btac715. [CrossRef] [PubMed]

23. Brixi, G.; Ye, T.; Hong, L.; Wang, T.; Monticello, C.; Lopez-Barbosa, N.; Vincoff, S.; Yudistyra, V.; Zhao, L.; Haarer, E.; et al. SaLT&PepPr is an interface-predicting language model for designing peptide-guided protein degraders. *Commun. Biol.* **2023**, *6*, 1081.
24. Romero-Molina, S.; Ruiz-Blanco, Y.B.; Mieres-Perez, J.; Harms, M.; Münch, J.; Ehrmann, M.; Sanchez-Garcia, E. PPI-affinity: A web tool for the prediction and optimization of protein–peptide and protein–protein binding affinity. *J. Proteome Res.* **2022**, *21*, 1829–1841. [CrossRef] [PubMed]
25. Chandra, A.; Sharma, A.; Dehzangi, I.; Tsunoda, T.; Sattar, A. PepCNN deep learning tool for predicting peptide binding residues in proteins using sequence, structural, and language model features. *Sci. Rep.* **2023**, *13*, 20882. [CrossRef]
26. Azim, S.M.; Sabab, N.H.N.; Noshadi, I.; Alinejad-Rokny, H.; Sharma, A.; Shatabda, S.; Dehzangi, I. Accurately predicting anticancer peptide using an ensemble of heterogeneously trained classifiers. *Inform. Med. Unlocked* **2023**, *42*, 101348. [CrossRef]
27. Wei, L.; Zhou, C.; Su, R.; Zou, Q. PEPred-Suite: Improved and robust prediction of therapeutic peptides using adaptive feature representation learning. *Bioinformatics* **2019**, *35*, 4272–4280. [CrossRef]
28. Kazmirchuk, T.D.D.; Bradbury-Jost, C.; Withey, T.A.; Gessese, T.; Azad, T.; Samanfar, B.; Dehne, F.; Golshani, A. Peptides of a feather: How computation Is taking peptide therapeutics under its wing. *Genes* **2023**, *14*, 1194. [CrossRef]
29. Yan, C.; Zou, X. Predicting peptide binding sites on protein surfaces by clustering chemical interactions. *Biophys. J.* **2015**, *108*, 215a. [CrossRef]
30. Jones, D.T. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* **1999**, *292*, 195–202. [CrossRef]
31. Heffernan, R.; Paliwal, K.; Lyons, J.; Dehzangi, A.; Sharma, A.; Wang, J.; Sattar, A.; Yang, Y.; Zhou, Y. Improving prediction of secondary structure, local backbone angles and solvent accessible surface area of proteins by iterative deep learning. *Sci. Rep.* **2015**, *5*, 11476. [CrossRef] [PubMed]
32. Yang, Y.; Heffernan, R.; Paliwal, K.; Lyons, J.; Dehzangi, A.; Sharma, A.; Wang, J.; Sattar, A.; Zhou, Y. SPIDER2: A package to predict secondary structure, accessible surface area, and main-chain torsional angles by deep neural networks. In *Prediction of Protein Secondary Structure*; Humana Press: New York, NY, USA, 2017; pp. 55–63.
33. Akbar, S.; Raza, A.; Zou, Q. Deepstacked-AVPs: Predicting antiviral peptides using tri-segment evolutionary profile and word embedding based multi-perspective features with deep stacking model. *BMC Bioinform.* **2024**, *25*, 102. [CrossRef]
34. Wardah, W.; Dehzangi, A.; Taherzadeh, G.; Rashid, M.A.; Khan, M.G.; Tsunoda, T.; Sharma, A. Predicting protein-peptide binding sites with a deep convolutional neural network. *J. Theor. Biol.* **2020**, *496*, 110278. [CrossRef] [PubMed]
35. Wang, R.; Jin, J.; Zou, Q.; Nakai, K.; Wei, L. Predicting protein–peptide binding residues via interpretable deep learning. *Bioinformatics* **2022**, *38*, 3351–3360. [CrossRef]
36. Shanker, S.; Sanner, M.F. Predicting protein–peptide interactions: Benchmarking deep learning techniques and a comparison with focused docking. *J. Chem. Inf. Model.* **2023**, *63*, 3158–3170. [CrossRef]
37. Yin, S.; Mi, X.; Shukla, D. Leveraging machine learning models for peptide–protein interaction prediction. *RSC Chem. Biol.* **2024**, *5*, 401–417. [CrossRef]
38. Devnath, L.; Fan, Z.; Luo, S.; Summons, P.; Wang, D. Detection and visualisation of pneumoconiosis using an ensemble of multi-dimensional deep features learned from Chest X-rays. *Int. J. Environ. Res. Public Health* **2022**, *19*, 11193. [CrossRef]
39. Mishra, R. Support Vector Machines Application for Prediction Binding Elements. In Proceedings of the 2024 IEEE International Conference on Big Data & Machine Learning (ICBDML), Bhopal, India, 24–25 February 2024; pp. 265–269.
40. Yuan, Q.; Yang, Y. Sequence-based predictions of residues that bind proteins and peptides. In *Machine Learning in Bioinformatics of Protein Sequences: Algorithms, Databases and Resources for Modern Protein Bioinformatics*; World Scientific Publishing: Singapore, 2023; pp. 237–263.
41. Ye, J.; Li, A.; Zheng, H.; Yang, B.; Lu, Y. Machine Learning Advances in Predicting Peptide/Protein-Protein Interactions Based on Sequence Information for Lead Peptides Discovery. *Adv. Biol.* **2023**, *7*, 2200232. [CrossRef] [PubMed]
42. Arif, M.; Fang, G.; Fida, H.; Musleh, S.; Yu, D.J.; Alam, T. iMRSAPred: Improved Prediction of Anti-MRSA Peptides Using Physicochemical and Pairwise Contact-Energy Properties of Amino Acids. *ACS Omega* **2024**, *9*, 2874–2883. [CrossRef]
43. Boltz, K.W.; Gonzalez-Moa, M.J.; Stafford, P.; Johnston, S.A.; Svarovsky, S.A. Peptide microarrays for carbohydrate recognition. *Analyst* **2009**, *134*, 650–652. [CrossRef]
44. Fu, J.; Reinhold, J.; Woodbury, N.W. Peptide-modified surfaces for enzyme immobilization. *PLoS ONE* **2011**, *6*, e18692. [CrossRef] [PubMed]
45. Taguchi, Y.H.; Gromiha, M.M. Application of amino acid occurrence for discriminating different folding types of globular proteins. *BMC Bioinform.* **2007**, *8*, 404. [CrossRef] [PubMed]
46. Dehzangi, A.; Sharma, A.; Lyons, J.; Paliwal, K.K.; Sattar, A. A mixture of physicochemical and evolutionary–based feature extraction approaches for protein fold recognition. *Int. J. Data Min. Bioinform.* **2015**, *11*, 115–138. [CrossRef]
47. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
48. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
49. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
50. Krissinel, E. On the relationship between sequence and structure similarities in proteomics. *Bioinformatics* **2007**, *23*, 717–723. [CrossRef]
51. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4765–4774.

52. Levy, J.J.; Titus, A.J.; Petersen, C.L.; Chen, Y.; Salas, L.A.; Christensen, B.C. MethylNet: An automated and modular deep learning approach for DNA methylation analysis. *BMC Bioinform.* **2020**, *21*, 108. [CrossRef]
53. Yap, M.; Johnston, R.L.; Foley, H.; MacDonald, S.; Kondrashova, O.; Tran, K.A.; Nones, K.; Koufariotis, L.T.; Bean, C.; Pearson, J.V.; et al. Verifying explainability of a deep learning tissue classifier trained on RNA-seq data. *Sci. Rep.* **2021**, *11*, 2641. [CrossRef]
54. 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.
55. Fan, A.; Jernite, Y.; Perez, E.; Grangier, D.; Weston, J.; Auli, M. ELI5: Long form question answering. *arXiv* **2019**, arXiv:1907.09190.
56. Akbar, S.; Ali, F.; Hayat, M.; Ahmad, A.; Khan, S.; Gul, S. Prediction of antiviral peptides using transform evolutionary & SHAP analysis based descriptors by incorporation with ensemble learning strategy. *Chemom. Intell. Lab. Syst.* **2022**, *230*, 104682.
57. Dickinson, Q.; Meyer, J.G. Positional SHAP (PoSHAP) for Interpretation of machine learning models trained from biological sequences. *PLoS Comput. Biol.* **2022**, *18*, e1009736. [CrossRef]
58. Prabhu, H.; Bhosale, H.; Sane, A.; Dhadwal, R.; Ramakrishnan, V.; Valadi, J. Protein feature engineering framework for AMPylation site prediction. *Sci. Rep.* **2024**, *14*, 8695. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Article

Classification and Regression of Pinhole Corrosions on Pipelines Based on Magnetic Flux Leakage Signals Using Convolutional Neural Networks

Yufei Shen and Wenxing Zhou *

Department of Civil & Environmental Engineering, The University of Western Ontario, London, ON N6A 5B9, Canada
* Correspondence: wzhou@eng.uwo.ca

Abstract: Pinhole corrosions on oil and gas pipelines are difficult to detect and size and, therefore, pose a significant challenge to the pipeline integrity management practice. This study develops two convolutional neural network (CNN) models to identify pinholes and predict the sizes and location of the pinhole corrosions according to the magnetic flux leakage signals generated using the magneto-static finite element analysis. Extensive three-dimensional parametric finite element analysis cases are generated to train and validate the two CNN models. Additionally, comprehensive algorithm analysis evaluates the model performance, providing insights into the practical application of CNN models in pipeline integrity management. The proposed classification CNN model is shown to be highly accurate in classifying pinholes and pinhole-in-general corrosion defects. The proposed regression CNN model is shown to be highly accurate in predicting the location of the pinhole and obtain a reasonably high accuracy in estimating the depth and diameter of the pinhole, even in the presence of measurement noises. This study indicates the effectiveness of employing deep learning algorithms to enhance the integrity management practice of corroded pipelines.

Keywords: deep learning algorithm; convolutional neural network; pipeline; corrosion; pinhole; finite element analysis; magnetic flux leakage signal

1. Introduction

Pipelines are economical and safe means for transporting and distributing large volumes of oil and gas products across great distances [1]. However, pipeline failures can lead to severe consequences such as fatalities, economic losses, and environmental damages [2]. Among the various failure mechanisms threatening the structural integrity of pipelines, corrosion is a leading threat [3]. The in-line inspection (ILI) tool is widely used for detecting and measuring corrosions to assess pipeline integrity [4], and magnetic flux leakage (MFL) is the predominant ILI technology for identifying internal and external corrosion defects in both liquid and gas pipelines [5]. The Pipeline Operators Forum (POF) suggests that a single corrosion anomaly on the internal or external surface of a pipeline can be categorized into one of seven classes based on the ILI-reported corrosion length and width as illustrated in Figure 1 [6], i.e., axial slotting, circumferential slotting, axial grooving, circumferential grooving, general, pinhole and pitting, to better quantify the accuracy of ILI tools and facilitate the selection of appropriate models to evaluate the burst capacity of pipelines containing corrosion defects [6,7].

The estimation of the corrosion anomaly sizes based on MFL signals falls into the inverse modeling realm. The pipeline industry frequently employs the iterative inverse model because of its remarkable precision [8–10]. However, the iterative inverse model is computationally costly. Inverse models based on machine learning provide promising alternatives to sizing corrosion defects from MFL signals with high accuracy but are more computationally efficient compared to the iterative inverse models [11–13]. For instance,

Kandroodi et al. [14] fed the algorithm-estimated defect width and length, as well as the extracted peak-to-peak values from the MFL signals, into a Gaussian radial basis function neural network to predict the corrosion depth. Feng et al. [15] proposed an error adjustment methodology for reconstructing the corrosion profiles from MFL signals based on a radial basis function neural network. However, these models have certain limitations: the training of these models relies on manually chosen features from the MFL signals, potentially omitting valuable data necessary for accurately predicting the corrosion profile. Additionally, the majority of these techniques focus solely on one MFL signal dimension, neglecting the critical information present in the other two dimensions.

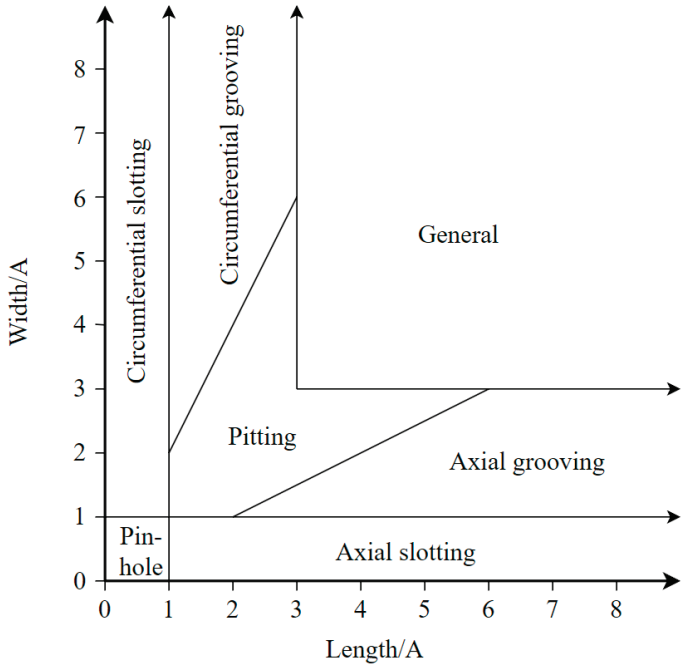


Figure 1. Seven corrosion anomaly categories are based on the anomaly length and width. Note: A = max{10 mm, pipe wall thickness}.

Over the past decade, convolutional neural networks (CNNs) have risen as a revolutionary deep learning algorithm, yielding remarkable results across various pattern recognition fields, including image and voice processing [16,17]. The CNN tool can automatically extract deep features from input data in place of manually designed feature extractors; therefore, it is well suited to solve the inverse modeling based on the MFL signals. The application of CNN in the inverse analysis of MFL signals has been reported in the literature. Lu et al. [18] introduced the visual transformation CNN, which predicts the length, width, and depth of anomalies by taking into account two components (i.e., circumferential and radial) of the MFL signal while disregarding the longitudinal component. Shen and Zhou [19] employed CNN to estimate the locations and dimensions of corrosions on steel pipelines based on three components of the MFL signals. Wang and Chen [20] proposed a CNN architecture comprising two main modules: one for anomaly classification and another for anomaly size regression. The input of the classification module consists of all three components of the MFL signal, and the outputs are seven anomaly categories, as illustrated in Figure 1. The regression module incorporates seven separate CNNs, each tailored to a specific anomaly type, predicting dimensions such as width, length, and depth of the defect.

According to a report from the European Gas Pipeline Incident Data Group (EGIG) [21], pinhole corrosion is identified as a major contributor to pipeline failures due to its potential to evolve from a small initial volume of metal loss into more severe metal loss, given the challenge in detection via pressure monitoring, particularly when the leaked products do not exceed ground level [22,23]. Despite the impact of pinholes on pipeline integrity,

studies that are specifically focused on detecting and sizing pinhole corrosions are limited in the literature, thus creating a knowledge gap. Furthermore, small corrosion anomalies such as pinholes may sometimes overlap with a relatively large corrosion area [23–25], creating a complex defect known in this study as a pinhole-in-general corrosion (PIC) defect. There are also few studies in the literature investigating the detection and sizing of PIC defects.

The objective of this study is to develop deep learning algorithms, i.e., CNN models, to classify and size pinholes and PIC defects based on a large set of MFL signals. The classification CNN model deals with six scenarios, i.e., pinholes on the internal or external pipe surface, general corrosions on the internal or external surface, and PIC defects on the internal or external surface. The signals for the pinholes and PIC defects identified by the classification model are then fed into the regression CNN model to predict the sizes and location of the pinhole and PIC defects. In practice, it is not necessary to apply the two CNN models sequentially. Depending on the specific application, data can be fed directly into the regression model if the classification has already been carried out through other means and the primary goal is to predict the size of the defect. Since real MFL signals are typically proprietary and not publicly accessible, the MFL signals used in this study are numerically generated by applying the three-dimensional (3D) magneto-static finite element analyses (FEA) using the commercial software COMSOL (Version 5.6). The three-dimensional (i.e., longitudinal, circumferential, and radial) MFL signal maps serve as inputs for training and validating CNN models to classify the defects and predict the sizes, as well as the circumferential and longitudinal positions, of pinholes on the pipeline. The impact of the measurement noises associated with the MFL signal on the accuracy of the regression CNN model is also investigated. This research fills a knowledge gap in the literature by applying CNN to detect, locate, and size pinholes and pinhole-in-corrosion defects. It is noted that accurately identifying the position of the pinhole within the general corrosion is crucial as the relative position of the pinhole within the general corrosion has a marked effect on the burst capacity of the PIC defect and, therefore, has strong implications for the accurate prediction of the burst capacity. This study demonstrates the effectiveness of CNN for the detection and sizing of pinhole corrosion and the viability of CNN for applications in pipeline integrity management practice.

The rest of the paper is structured as follows. Section 2 introduces the principles of the MFL technology and essential background information on the relationship between MFL signals and anomaly features. Section 3 introduces the proposed FEA model and the simulation parameters of MFL signals. Section 4 describes the structure of the developed CNN models and their application to the classification and regression of pinholes. The corresponding predictive accuracy of the CNN models is also discussed in Section 4. Section 5 summarizes the concluding remarks of the study.

2. Principles of MFL Technique

The magneto-statics FEA is employed to numerically generate the MFL signals in the study. The corresponding governing equation derived from Maxwell's equations is shown in Equation (1):

$$\nabla \times \left(\frac{1}{\mu} \nabla \times \boldsymbol{A} \right) = 0 \tag{1}$$

where $\nabla\times$ denotes the curl operation; μ represents the magnetic permeability of the ferromagnetic specimen (N/A^2), and $\boldsymbol{A}$ is the magnetic potential vector. Equation (1) is solved under the specified boundary conditions to obtain $\boldsymbol{A}$, which is then used to obtain the magnetic flux density $\boldsymbol{B}$ as follows [26,27]:

$$\boldsymbol{B} = \nabla \times \boldsymbol{A} \tag{2}$$

The MFL technique relies on the measurement principle where a ferromagnetic material demonstrates MFL in the vicinity of a defect, as illustrated in Figure 2 [18]. The

measurement procedure is as follows: during the ILI process, the MFL tool is propelled through the pipeline, and magnets within the tool generate a magnetic field. In a perfect pipe, all magnetic flux remains within the pipe wall [28], resulting in a uniform magnetic field distribution [29]. However, in the presence of defects, some magnetic flux "leaks" outside the pipe, forming a magnetic leakage field close to the defect [30]. This leakage field is detected by Hall sensors, generating electrical signals [29]. The strength and distribution of the flux leakage vary depending on the geometry of the defect on the pipeline. By capturing the MFL signal, which typically consists of a three-dimensional magnetic flux density ($\boldsymbol{B}$) vector field, represented by $\boldsymbol{B}_x$ in the circumferential direction of the pipe, $\boldsymbol{B}_y$ in the radial direction, and $\boldsymbol{B}_z$ in the longitudinal (axial) direction, it is possible to estimate the size and location characteristics of the corrosion defect.

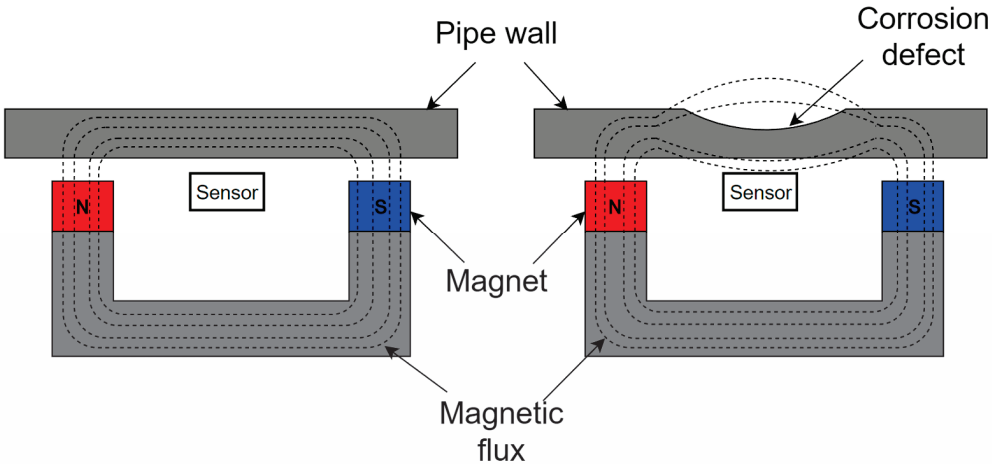


Figure 2. Principle of MFL technique.

Axial (AMFL) and circumferential (CMFL) MFL tools are the two types of MFL tools categorized by the arrangement of the magnets inside the MFL tool. In AMFL tools, the permanent magnets are oriented parallel to the longitudinal axis of the pipe. These tools excel in detecting and measuring defects in the circumferential direction of the pipe but are less effective in sizing features oriented axially [31]. Conversely, CMFL tools have magnets positioned in the circumferential direction of the pipeline. They are more precise in measuring features aligned along the longitudinal direction compared to AMFL tools. The CMFL tool is the main focus of this study because it is more accurate than the AMFL tool for sizing longitudinally-oriented corrosion anomalies, which are perpendicular to the hoop stress resulting from the pipe's internal pressure and, therefore, can have a great influence on the pipeline burst capacity.

3. Simulating MFL Signals Using FEA

3.1. Simulation Parameters

The CMFL tool considered in this study includes three pairs of permanent magnets. NdFeB is selected as the magnetic material with a high coercive force of 895,000 A/m, stable magnetic properties, and high magnetic energy [32]. The steel used to simulate the pipe is assumed to be grade X52, characterized by a remanent flux density (B_r) of 1T, an electrical conductivity of 5.882×10^6 S/m, coercive field strength (H_c) of 415 A/m, and saturation flux density (B_s) of 1.9T. Figure 3 shows the schematics of a cross-section of the CMFL tool and pipe section, and the geometric and material properties of each component are summarized in Table 1. It is noted that the lift-off value is assumed to be 5 mm, which is the distance between the sensor and the pipe's internal surface [33].

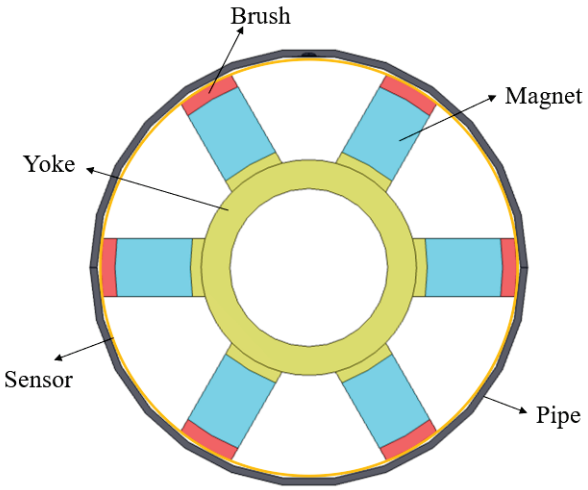


Figure 3. Illustration of CMFL tool.

Table 1. Geometry of every component within the proposed 3D CMFL model.

Elements	Pipe Section	Magnets	Brushes	Yoke
geometry (mm)	600 × 610 × 10 [a]	300 × 81 × 105 [b]	300 × 81 × 20 [b]	300 × 300 × 40 [a]

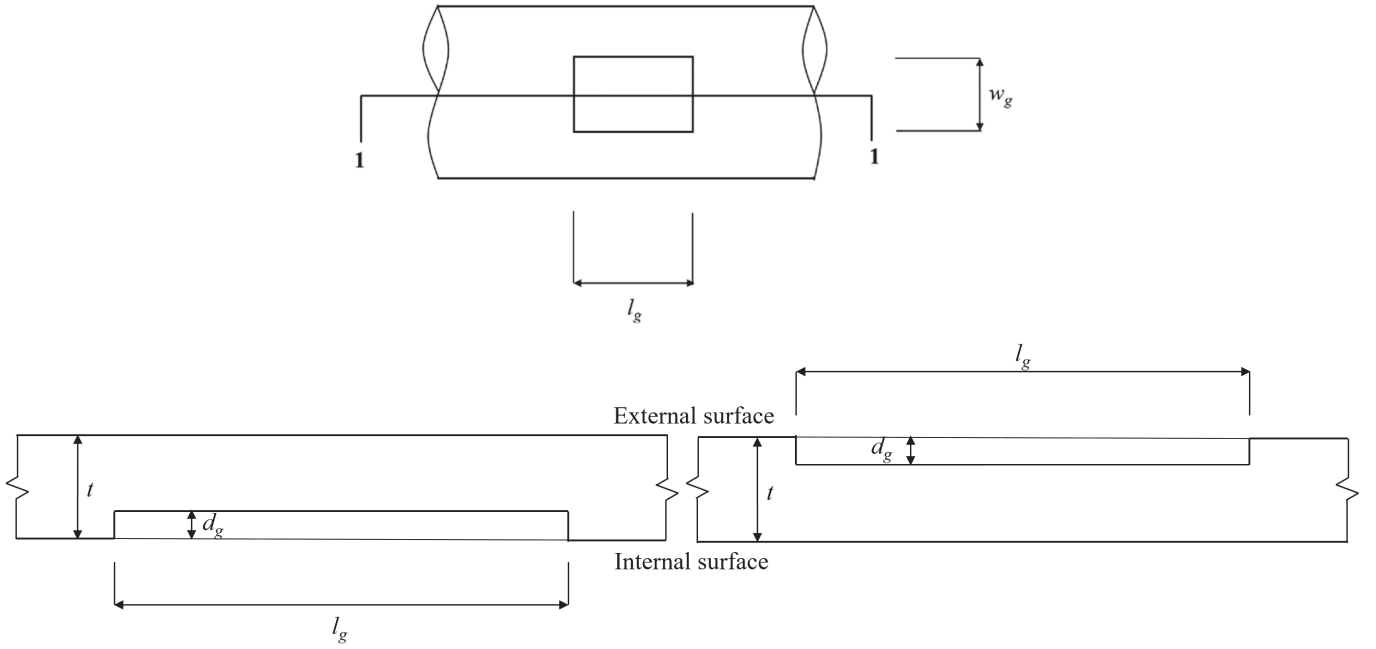
[a] Length × outside diameter × wall thickness. [b] Length × width × thickness.

3.2. FEA Cases

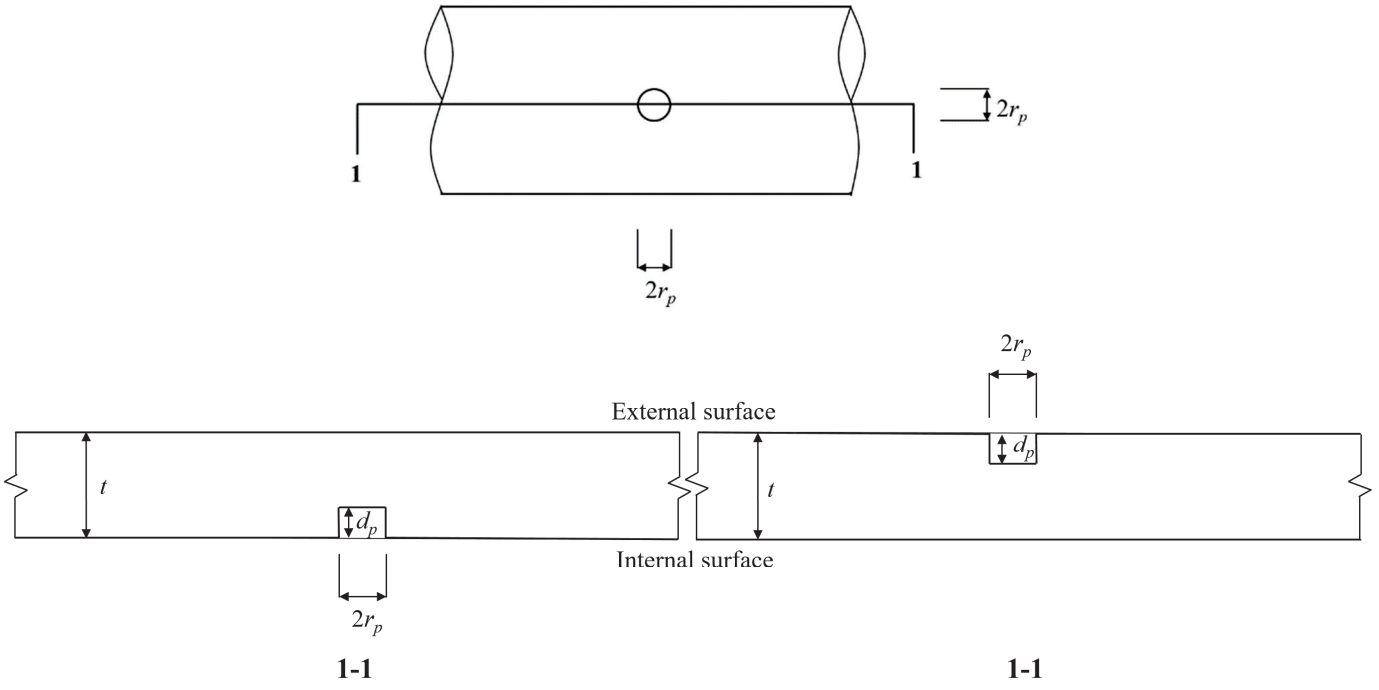
A series of FEA models are created and analyzed with the same magnetization conditions and pipe attributes, but they vary in corrosion types, geometric parameters, and locations in this study. The simulation involves 9600 analysis cases, with 1600 cases allocated to each of the six scenarios as illustrated in Figure 4, namely internal general corrosion (G_{in}), external general corrosion (G_{ex}), internal pinhole (P_{in}), external pinhole (P_{ex}), internal PIC defect (PIC_{in}), and external PIC defect (PIC_{ex}). The general corrosion is idealized as cuboidal-shaped, defined by the width (w_g), depth (d_g), and length (l_g), and the pinhole is idealized as cylindrical-shaped, defined by the radius (r_p) and depth (d_p). Note the center of the pinhole in a PIC defect is assumed to be coincident with that of the general corrosion, which is defined by two coordinates, i.e., its longitudinal location (h) and circumferential location (ϕ), within a 160 × 160 mm designated area as shown in Figure 5. This area is marked in light blue, with its center specified by the coordinates (r, $\varphi = 0$, $h = 0$), where r represents the inner radius of the pipe. This specific 160 × 160 mm square area on the inner pipe wall was selected because it encompasses the largest area, showing a consistent magnetic flux density profile between the neighboring magnets.

The analysis cases involving general corrosion only, i.e., G_{in} and G_{ex}, include four values of w_g, i.e., 30, 50, 70, and 90 mm, four values of d_g/t, i.e., 20, 40, 60, and 80%, where t denotes the pipe wall thickness, and four values of l_g, i.e., 30, 50, 70, and 90 mm. This results in 64 general corrosions in total with varying sizes. In terms of the corrosion location, five values of h (−50, −25, 0, 25, and 50 mm) and five values of ϕ (−10, −5, 0, 5, and 10 degrees) are considered. The permutations of 64 defects and 25 locations lead to 1600 cases for each of G_{in} and G_{ex}.

The analysis cases involving pinhole corrosion only, i.e., P_{in} and P_{ex}, include eight values of r_p, i.e., 1, 1.5, 2, 2.5, 3, 3.5, 4, and 4.5 mm, and eight values of d_p/t, i.e., 10, 20, 30, 40, 50, 60, 70, and 80%. This results in 64 pinhole corrosions in total with varying sizes. In terms of the corrosion location, five values of h (−50, −25, 0, 25, and 50 mm) and five values of ϕ (−10, −5, 0, 5, and 10 degrees) are considered. The permutation of 64 defects and 25 locations also leads to 1600 cases for each of P_{in} and P_{ex}.

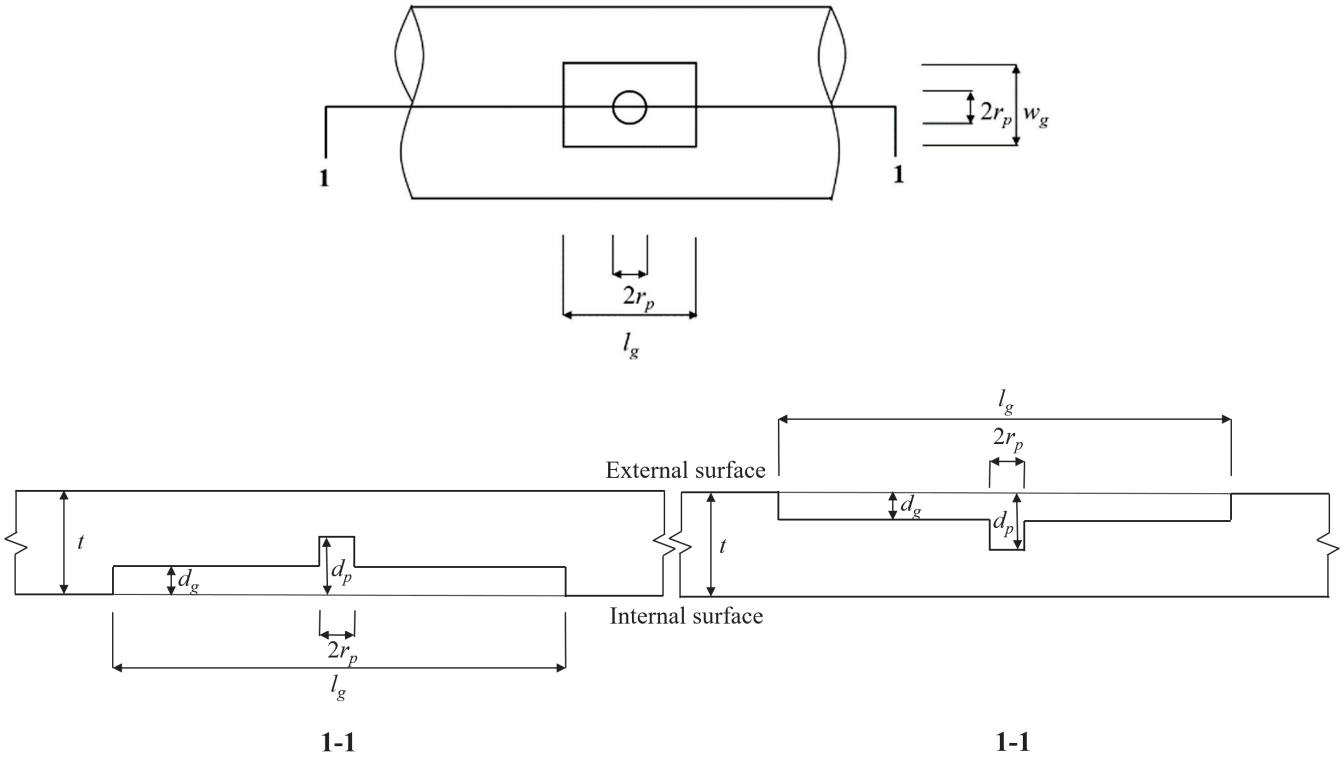


(**a**) Gin and Gex



(**b**) Pin and Pex

Figure 4. *Cont*.



(**c**) PIC_{in} and PIC_{ex}

Figure 4. Six types of corrosion situations: (**a**) G_{in} and G_{ex}; (**b**) P_{in} and P_{ex}; (**c**) PIC_{in} and PIC_{ex}.

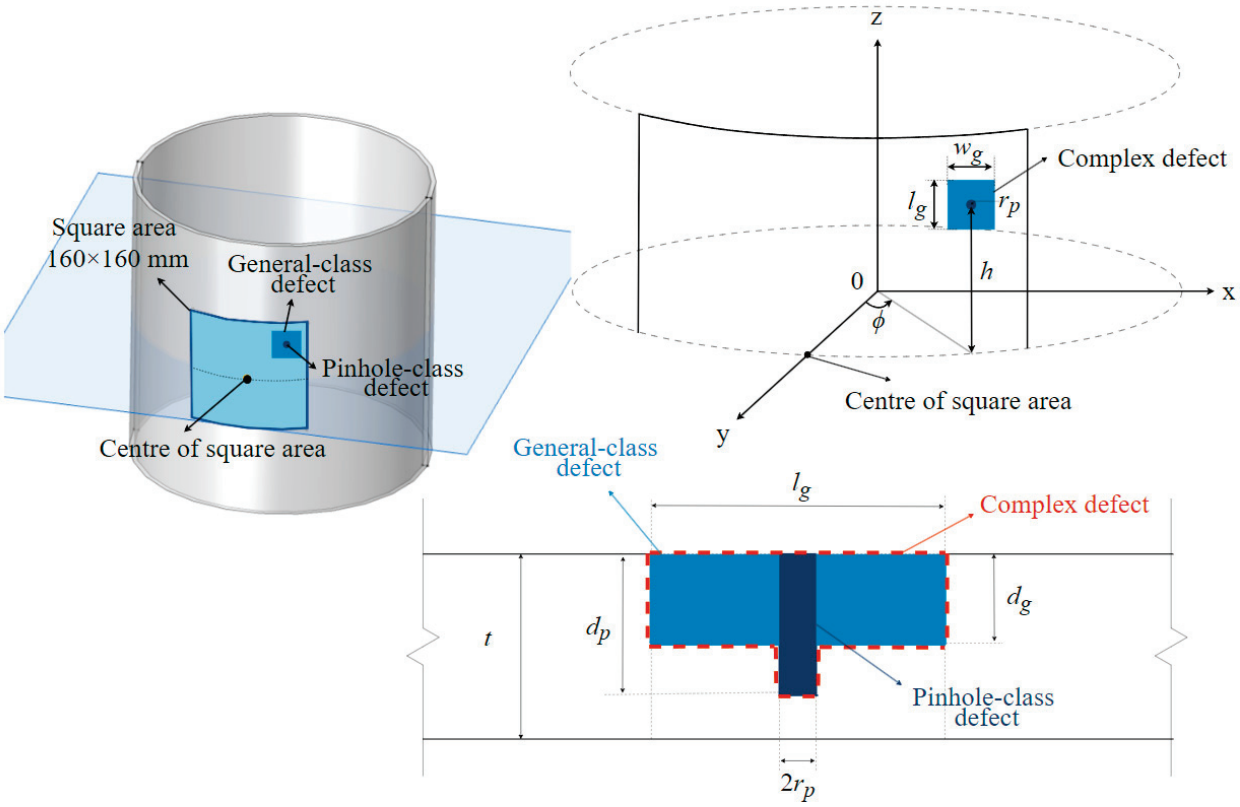


Figure 5. Cylindrical coordinate system defining location parameters h and ϕ.

Analysis cases involving PIC defects, i.e., PIC_{in} and PIC_{ex}, include $w_g = l_g = 50$ mm and four values of r_p, i.e., 1, 2, 3, and 4 mm. In terms of the sizes of general and pinhole corrosions, three groups are considered to capture a wider range of relative relationships between general and pinhole corrosions. The first includes d_g/t equal to 20% and d_p/t equal to 30, 40, 50, 60, 70, or 80%. The second group includes d_g/t equal to 30% and d_p/t equal to 40, 50, 55, 60, 70, or 80%. The last group includes d_g/t equal to 40% and d_p/t equal to 50, 60, 70, or 80%. It should be noted that d_p/t is assumed to be at least 10% larger than d_g/t in all three groups. This results in 64 PIC defects in total with varying sizes. In terms of the corrosion location, five values of h (−50, −25, 0, 25, and 50 mm) and five values of ϕ (−10, −5, 0, 5, and 10 degrees) are considered. The permutation of 64 defects and 25 locations leads to 1600 cases for each of PIC_{in} and PIC_{ex}. It follows that a dataset consisting of 9600 cases (1600 × 6 = 9600) is generated in this study. This large dataset is intended to facilitate the training and validation of the CNN model, enabling comprehensive analysis and robust model development.

The developed FEA model meshes using 4-node tetrahedral elements, where the minimum size is 4 mm, and the maximum mesh size is 55 mm after a convergence study. Specifically, grids inside the 160 mm × 160 mm area are finely resolved at 1 mm intervals. At each grid point, the value of $\boldsymbol{B}$ is calculated via interpolation between two mesh nodes. The system of nonlinear equations is solved using the Newton-Raphson method. As an illustration, Figure 6 depicts the 3D magnetic flux density information (i.e., B_x, B_y, and B_z) within the 160 mm × 160 mm area of a selected FE model. The proposed FEA model was validated by comparing the simulation results with experimental data from a CMFL tool reported by Ireland and Torres [34] and an AMFL tool reported by Azizzadeh and Safizadeh [35]. Readers are referred to Shen and Zhou [19] for details of the validation.

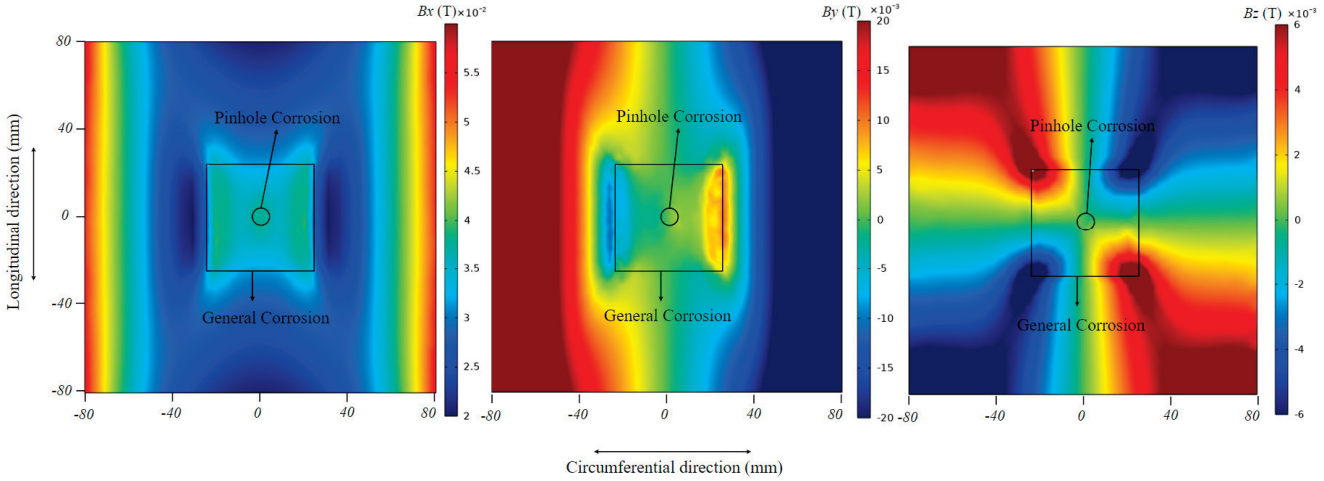


Figure 6. FEA-obtained B_x, B_y, and B_z corresponding to a PIC defect on the internal pipe surface (w_g = 50 mm, l_g = 50 mm, d_g/t = 40%, r_p = 4 mm, d_p/t = 80%, h = 0 mm and ϕ = 0 degree).

MFL signals in practical applications inevitably include noise stemming from factors such as non-uniform pipe wall thickness and differences in sensor lift-off values [36]. These noises can impact the accuracy of the inverse model, necessitating the inclusion of noises in the FEA-simulated MFL signals. The signal-to-noise (SNR) ratio serves as an important parameter for quantifying the noise in the three-dimensional MFL signal. Equation (3) defines SNR in this study [37]:

$$SNR = 10\log_{10}\left(\frac{\sum_{n=1}^{160}\sum_{m=1}^{160}\left[B_x(n,m)^2 + B_y(n,m)^2 + B_z(n,m)^2\right]}{\sum_{n=1}^{160}\sum_{m=1}^{160}\left[w_x(n,m)^2 + w_y(n,m)^2 + w_z(n,m)^2\right]}\right) \quad (3)$$

where w_x, w_y and w_z denote the noises along three directions of the pipe, namely circumferential, radial and longitudinal directions, respectively, and (n, m) (where n, m = 1, 2,. . ., 160) represents the coordinate of each grid (with 1 mm resolution) in the 160 mm × 160 mm area, with a total of 25,600 grids. By considering a representative value of SNR equal to 20 [38], three 160 × 160 matrices of Gaussian-distributed white noises, which are assumed to be independent of each other, are simulated and then applied to B_x, B_y, and B_z, respectively. Each matrix assumes a mean value of zero, with standard deviations adjusted through trial and error until achieving the predefined signal-to-noise ratio. Let σ_n denote the standard deviation of w_x; the standard deviations of w_y and w_z are assumed to be $0.75\sigma_n$ and $0.25\sigma_n$, respectively. The scaling of σ_n is based on the relative magnitudes of B_x, B_y, and B_z derived from the finite element analysis: for a given analysis case, B_x consistently has the highest magnitude, followed by B_y and B_z, respectively. The final noisy input to the CNN model consists of the three matrices B_x, B_y, and B_z, each combined with respective noisy matrices w_x, w_y, and w_z, resulting in noisy representations of the MFL signals.

It is emphasized that the noisy MFL signal matrices are only fed into the regression CNN model as an extended part in Section 4.2.4. Fan et al. [39] reported that the effect of Gaussian noise on the CNN classification model accuracy is minor. Acharya et al. [40] indicated that removing the noise is not necessary for image classification with deep learning algorithms [41]. Therefore, we only consider the impact of the noises on the accuracy of the regression model in the present study.

4. Convolutional Neural Network

4.1. Classification CNN

4.1.1. Input Information

The input to a CNN model is commonly represented as an RGB color image, including three color channels corresponding to green, red, and blue, respectively. This input is compatible with the three-dimensional MFL signals. For a given FE model, the 3D magnetic flux density information (i.e., B_x, B_y, and B_z) within the designated square area is exported from the COMSOL software at the 160 mm × 160 mm area grids with a resolution of 1 mm. The three generated 160 × 160 matrices of noise-free MFL signals (one matrix for each dimension) are denoted as the input of the proposed CNN model, and the six types of corrosion (i.e., G_{in}, G_{ex}, P_{in}, P_{ex}, PIC_{in}, and PIC_{ex}) are the output.

4.1.2. Proposed Structure

The proposed classification CNN model in this study comprises a total of 20 layers: layers 1 to 12 are dedicated to feature extraction, while layers 13 to 20 are focused on classification. Layer 20 serves as the output layer, comprising six components that represent the six corrosion types. The detailed information on each layer is summarized in Table 2. As shown in Table 2, the size of the filter in the convolution layer is selected to be (5 × 5), which is a typical size for large filters [42]. The size of the filter in the maxpooling layer is (2 × 2), consistent with those commonly used in the existing literature [43,44]. Several dropout layers that randomly drop a certain percentage of neurons from the neural network during the training process are included in the model to avoid overfitting. The employed dropout rate is 0.2, i.e., 20% of the neurons in the neural network are randomly deactivated. Figure 7 depicts the architecture of the proposed classification model with the dimension of each layer, where the input, convolution layers, and max-pooling layers are depicted in pink, blue, and green, respectively. Note that the dropout layers are not shown in Figure 7 as they do not impact the data dimensionality. The thick white line represents the flattened layer, the fully connected layers are represented in orange, and the output layers are shown in yellow in Figure 7.

Table 2. CNN classification model layer information.

Part	Layer No.	Layer Name	Parameters
Feature extraction	1	Convolution	64 filters with size equals to (5 × 5)
	2	Maxpooling	Pooling size equals to (2 × 2)
	3	Dropout	Rate = 0.2
	4	Convolution	128 filters with size equals to (5 × 5)
	5	Maxpooling	Pooling size equals to (2 × 2)
	6	Dropout	Rate = 0.2
	7	Convolution	256 filters with size equals to (5 × 5)
	8	Maxpooling	Pooling size equals to (2 × 2)
	9	Dropout	Rate = 0.2
	10	Convolution	512 filters with size equals to (5 × 5)
	11	Maxpooling	Pooling size equals to (2 × 2)
	12	Dropout	Rate = 0.2
Classification	13	Flatten layer	Unit number equals to 51,200
	14	Fully connected layer	Unit number equals to 128
	15	Dropout	Rate = 0.2
	16	Fully connected layer	Unit number equals to 64
	17	Dropout	Rate = 0.2
	18	Fully connected layer	Unit number equals to 32
	19	Dropout	Rate = 0.2
	20	Output 1	Unit number equals to 1
		Output 2	Unit number equals to 1
		Output 3	Unit number equals to 1
		Output 4	Unit number equals to 1
		Output 5	Unit number equals to 1
		Output 6	Unit number equals to 1

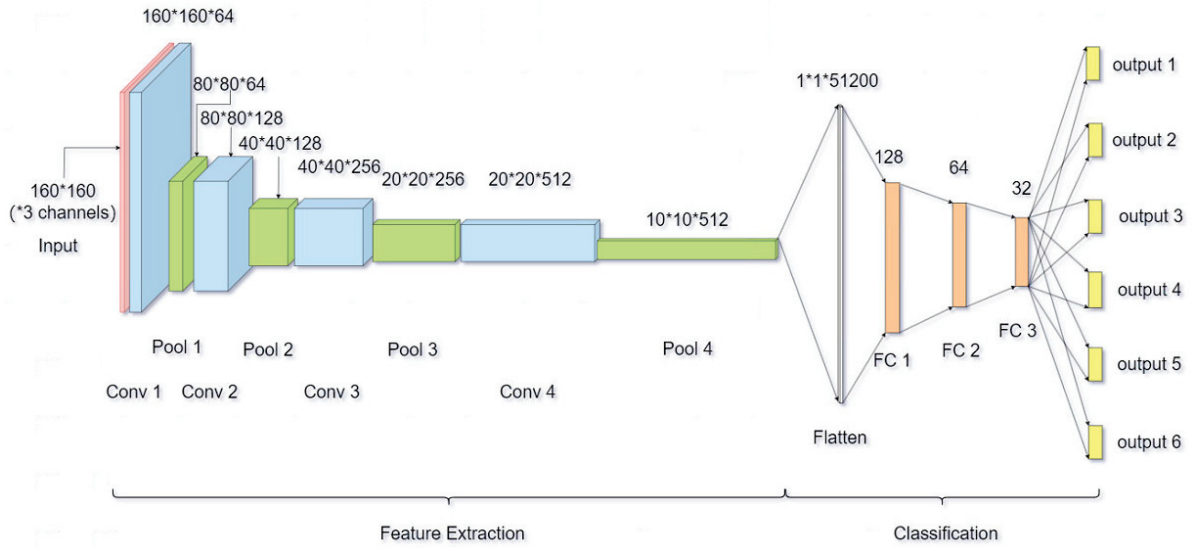


Figure 7. Proposed CNN classification model structure.

4.1.3. Results

By adopting an 80–20% training-test data ratio, there are 1280 training cases and 320 test cases for each of the six corrosion types. The classification results obtained from the proposed CNN model are summarized in Table 3. Table 3 demonstrates that the proposed model consistently achieves prediction accuracies higher than 0.93 in both the training and test datasets, with the lowest accuracy of 0.934 observed in the training dataset for G_{ex}. It is noted that the model performs better in the test dataset than the training dataset, with accuracy rates of each corrosion type in the test dataset higher than or equal to the rates in the training dataset. This may be due to the regularization techniques, such as dropout, adopted in the CNN model. Although these techniques can prevent overfitting, they may also introduce noise or uncertainty during training, which could temporarily degrade the model's performance on the training data while improving the generalization and performance on the test data [45].

Table 3. Results of the classification CNN model.

Category No.	Category	Total	Train	Test	Misprediction in Train	Misprediction in Test	Train Accuracy	Test Accuracy
0	G_{in}	1600	1280	320	7	0	0.978	1.000
1	G_{ex}	1600	1280	320	21	0	0.934	1.000
2	P_{in}	1600	1280	320	0	0	1.000	1.000
3	P_{ex}	1600	1280	320	2	0	0.994	1.000
4	PIC_{in}	1600	1280	320	1	0	0.997	1.000
5	PIC_{ex}	1600	1280	320	1	1	0.997	0.997

Notably, 32 cases in the training dataset and one case in the test dataset have been misclassified, resulting in a misprediction rate of 0.34% (33/9600 = 0.34%), demonstrating the excellent accuracy of the developed model. Figure 8 illustrates the distribution of mispredictions in the training dataset across different categories. For instance, 65.6% (i.e., 21) of misclassified training cases belong to the G_{ex} category. Among these, four cases are misclassified as $P_{in,}$ and 17 cases are misclassified as P_{ex}. Furthermore, six out of the 33 cases have misclassified locations (internal/external surface of the pipe), i.e., two P_{ex} cases misclassified as $P_{in,}$ and four G_{ex} cases misclassified as P_{in}.

Note that the model achieves better classification accuracy for P_{in}, P_{ex}, $PIC_{in,}$ and PIC_{ex} than for G_{in} and G_{ex}. The values of d_g/t included in the entire dataset for all six types of corrosions are 20, 30, 40, 60, and 80%, while the values of d_p/t included in the entire dataset are 20, 30, 40, 50, 55, 60, 70 and 80%. The wider range of the corrosion depth for cases involving pinholes (i.e., P_{in}, P_{ex}, PIC_{in}, and PIC_{ex}) compared to the cases involving general corrosion (G_{in} and G_{ex}) allows the model to learn a richer set of features and variations associated with pinhole corrosion, making the model more adept at distinguishing between different types and severity of pinhole corrosion. All of the 28 misclassified cases in the G_{in} and G_{ex} categories involve relatively shallow corrosions, among which 26 cases have d_g/t = 20% and two cases have d_g/t = 40%. The distribution of the 28 misclassified cases by w_g/l_g is shown in Figure 9. Figure 9 indicates that the classification CNN model is less accurate for wide general corrosions with w_g/l_g larger than 1.67. It is also noteworthy that the developed model is less accurate for cases containing external corrosions (i.e., G_{ex}, P_{ex}, and PIC_{ex}) than for cases involving internal corrosions (i.e., G_{in}, P_{in}, and PIC_{in}). A potential explanation is that the model input (i.e., the MFL signal map) for external defects has less pronounced features because the variation in $\boldsymbol{B}$ due to the metal-loss defect on the external pipe surface is smaller than that caused by the internal metal-loss defect as the external defect is farther away from the sensor than the internal defect.

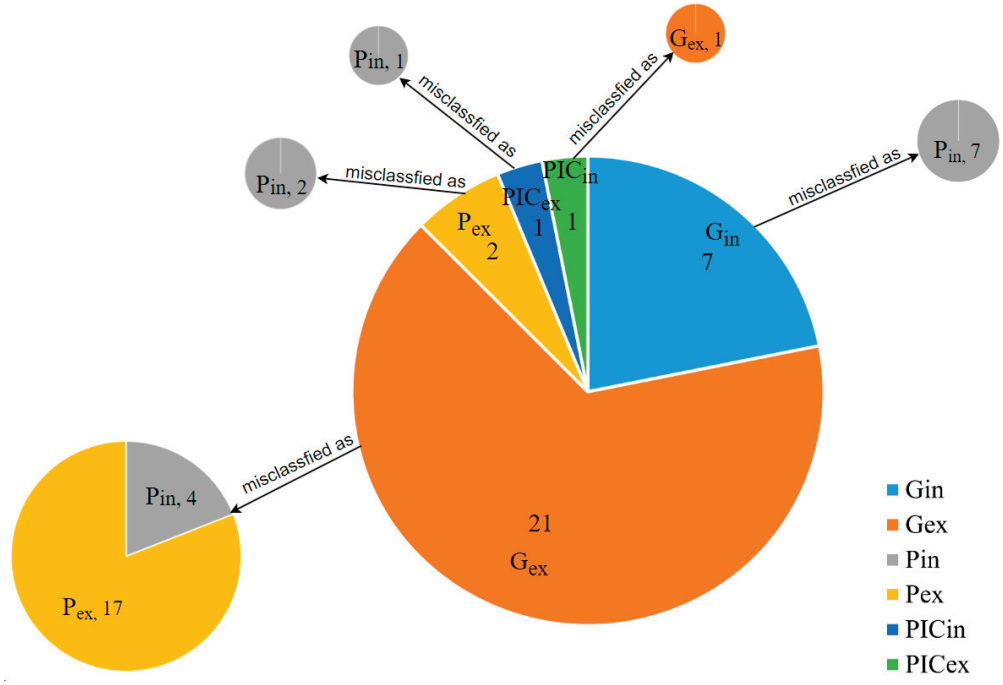


Figure 8. Distribution of misclassifications in the training dataset.

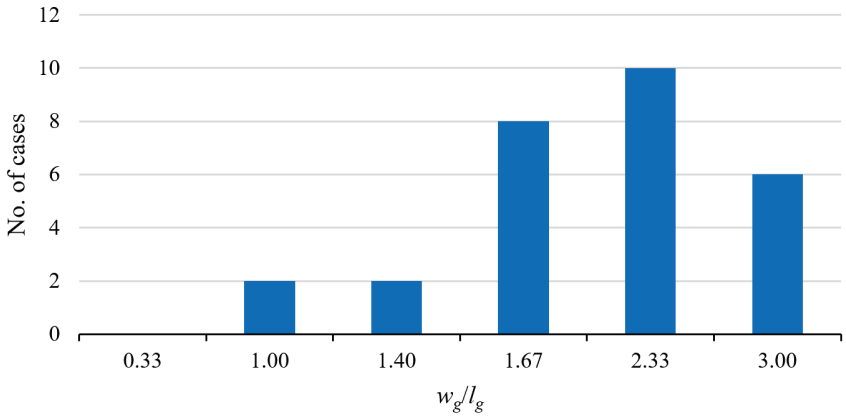


Figure 9. Distribution of misclassified cases in G_{in} and G_{ex} by w_g/l_g.

4.2. Regression CNN

4.2.1. Input Information

The results of the classification CNN model are utilized to extract correctly classified cases for P_{in}, P_{ex}, PIC_{in}, and PIC_{ex}, excluding four misclassifications in the training dataset and one misclassification in the test dataset, resulting in 1600, 1598, 1599, and 1598 input cases for P_{in}, P_{ex}, PIC_{in}, and PIC_{ex}, respectively, to train and validate the regression CNN model. The three generated 160 × 160 matrices of noise-free MFL signals for each of the 6395 (=1600 + 1598 + 1599 + 1598) cases are the input to the developed regression model, and the predicted r_p, d_p/t, h, and ϕ values for each case are the output. The correctly classified data are adopted to train the regression CNN model because including misclassified data (such as a general corrosion anomaly misclassified as a pinhole) in training is problematic, e.g., defining an error function that is applicable to all input data, including misclassified data. Alternatively, the training of the regression CNN model can be completely separated from the classification model; that is, the entire 6400 parametric finite element analysis cases involving pinholes can be employed to train the regression model. In this study, the correctly classified dataset, which consists of 6395 cases and is almost identical to the complete dataset, is employed to train and validate the regression CNN model.

4.2.2. Proposed Structure

The proposed regression CNN model in the present study comprises a total of 23 layers: layers 1 to 16 are dedicated to feature extraction, while layers 17 to 23 are focused on regression. Layer 23 is the output layer, which includes four components representing the predicted r_p, d_p/t, h, and ϕ values, respectively. The detailed CNN regression model layer information is presented in Table 4. It is noted that, in addition to the dropout layers introduced in Section 4.1.2, several batch normalization layers are incorporated within the typical CNN layers to prevent overfitting [46–48]. As shown in Table 4, the tuned hyperparameters in the feature extraction process of the regression model, such as the filter size of the convolution layer, the pooling size, and the dropout rate, are the same as those in the classification CNN model, while the employed dropout rate in the regression process of the regression CNN model is 0.5. Figure 10 depicts the architecture of the developed regression model with the dimensions of different layers. Note that the dropout and batch normalization layers are excluded from the diagram as they do not impact the data dimensionality. The color scheme utilized to distinguish various layers in Figure 10 is consistent with those utilized in Figure 7.

Table 4. CNN regression model layer information.

Part	Layer No.	Layer Name	Parameters
Feature extraction	1	Convolution	64 filters with size equals to (5 × 5)
	2	Batch normalization	
	3	Maxpooling	Pooling size equals to (2 × 2)
	4	Dropout	Rate = 0.2
	5	Convolution	128 filters with size equals to (5 × 5)
	6	Batch normalization	
	7	Maxpooling	Pooling size equals to (2 × 2)
	8	Dropout	Rate = 0.2
	9	Convolution	256 filters with size equals to (5 × 5)
	10	Batch normalization	
	11	Maxpooling	Pooling size equals to (2 × 2)
	12	Dropout	Rate = 0.2
	13	Convolution	512 filters with size equals to (5 × 5)
	14	Batch normalization	
	15	Maxpooling	Pooling size equals to (2 × 2)
	16	Dropout	Rate = 0.2
Regression	17	Flatten layer	Unit number equals to 51,200
	18	Dropout	Rate = 0.5
	19	Fully connected layer	Unit number equals to 64
	20	Fully connected layer	Unit number equals to 128
	21	Fully connected layer	Unit number equals to 256
	22	Fully connected layer	Unit number equals to 512
	23	Output 1	Unit number equals to 1
		Output 2	Unit number equals to 1
		Output 3	Unit number equals to 1
		Output 4	Unit number equals to 1

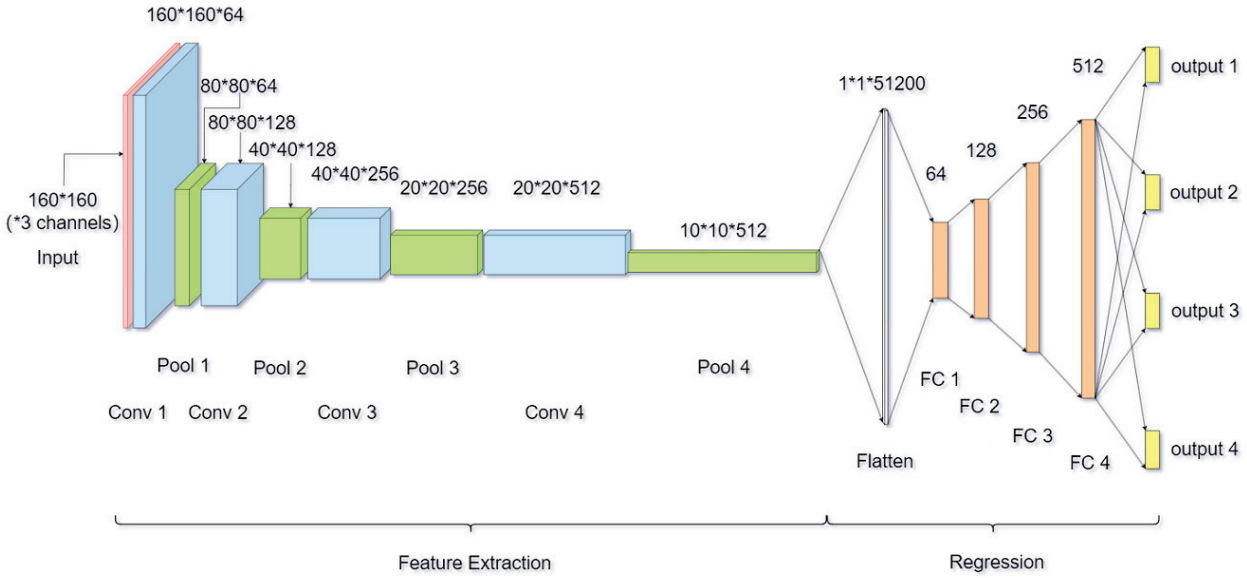


Figure 10. Proposed CNN regression model structure.

4.2.3. Results

The coefficient of determination (i.e., R^2) is used to measure the accuracy of each parameter prediction (i.e., r_p, d_p/t, h, and ϕ), defined by the following equation.

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (Y_{true,i} - Y_{pred,i})^2}{\sum_{i=1}^{N} (Y_{true,i} - \overline{Y}_{true})^2} \quad (4)$$

where $Y_{true,i}$ and $Y_{pred,i}$ represent the true and predicted values, respectively, of each predicted parameter for the ith (i = 1, 2,. . ., N) point in a given dataset containing N data points, and $\overline{Y}_{true}$ denotes the true mean value of N data points associated with a predicted parameter. Table 5 summarizes the R^2 values for the four parameters in the test dataset included in the regression model. The R^2 values are also plotted in a so-called radar chart in Figure 11. The true values of the four parameters r_p, d_p/t, h, and ϕ compared with the estimated values by the CNN for the corrosions in the test set are depicted in Figure 12. It is evident from Table 5 that the CNN model's predictions of the corrosion location parameters (i.e., h and ϕ) show strong agreement with the actual values, as reflected by the R^2 values of 1.0 for all four corrosion categories. In terms of the size predictions (i.e., r_p and d_p/t), the model is more accurate for internal corrosions (P_{in} and PIC_{in}) than for external corrosions (P_{ex} and PIC_{ex}). These results also indicate that the performance of the model is better for cases containing pinholes only than for cases containing PIC. This is expected as PIC introduces additional complexity, including the presence of general corrosion, which may impact the model's accuracy. Overall, the obtained R^2 values demonstrate the CNN model's accuracy in predicting the size parameters, with a higher accuracy observed for corrosions on the internal surface and pinholes. However, note that although the predictions for r_p and d_p/t are, in general, accurate, Figure 12 reveals relatively large error bands in both r_p and d_p/t predictions. For instance, the predicted values of r_p range from 1.14 mm to 3.45 mm for $(r_p)_{true}$ equal to 2 mm. The large error bands may be explained by the small dimensions of the pinhole corrosion and the limited spatial resolution of the model. Furthermore, the presence of composite corrosion features can introduce additional complexities and uncertainties, leading to larger error bands in the predictions. Therefore, further investigation is required to explore the factors contributing to the reduction of regression errors for pinhole corrosions.

Table 5. Results of the regression CNN model without noise.

Category No.	Category	Cases			Test R^2			
		Total	Train	Test	r_p	d_p/t	h	ϕ
0	P_{in}	1600	1280	320	0.91	0.95	1.00	1.00
1	P_{ex}	1598	1277	321	0.88	0.92	1.00	1.00
2	PIC_{in}	1599	1280	319	0.89	0.85	1.00	1.00
3	PIC_{ex}	1598	1279	319	0.86	0.77	1.00	1.00
Overall		6395	5116	1279	0.89	0.91	1.00	1.00

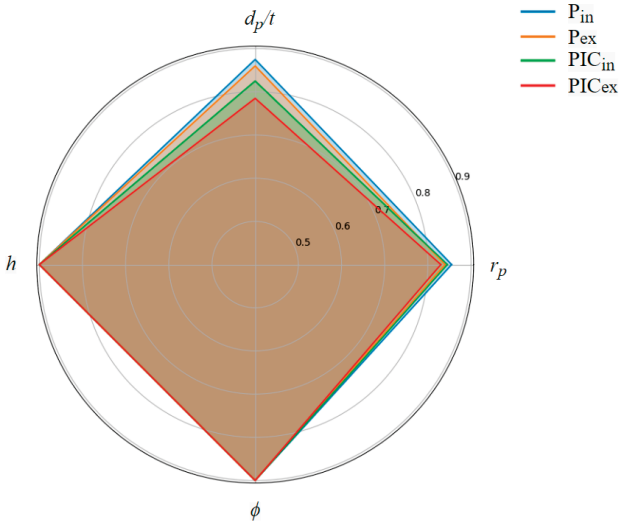


Figure 11. Radar chart for R^2 of metrics (r_p, d_p/t, h, and ϕ) for P_{in}, P_{ex}, PIC_{in}, and PIC_{ex} corrosions in the test dataset.

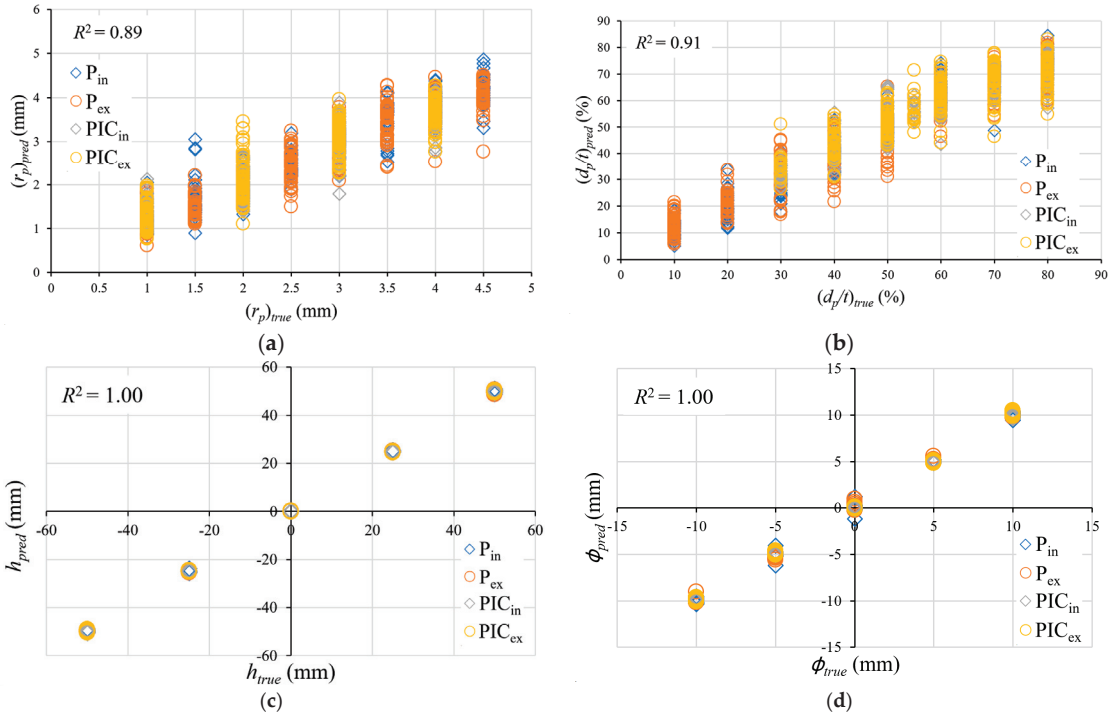


Figure 12. Comparison of true and predicted values of r_p, d_p/t, h, and ϕ for the anomalies in the regression test dataset without noise.

4.2.4. Influence of Noise

Figure 13 compares the results of the regression CNN model for noisy input (SNR = 20) with those of the noise-free scenario described in Section 4.2.3. With the presence of noise, the R^2 values for predicting r_p and d_p/t decrease somewhat compared to the noise-free scenario. The decrease in R^2 values is more prominent for P_{ex} and PIC_{ex} cases, indicating that the model is more affected by noise when predicting the sizes of external corrosion. Despite the decrease in the accuracy in predicting the corrosion sizes, the model maintains R^2 = 1 for the location parameters (h and ϕ) across all corrosion types except for P_{ex}, indicating its robustness in capturing the spatial information even in the presence of noise.

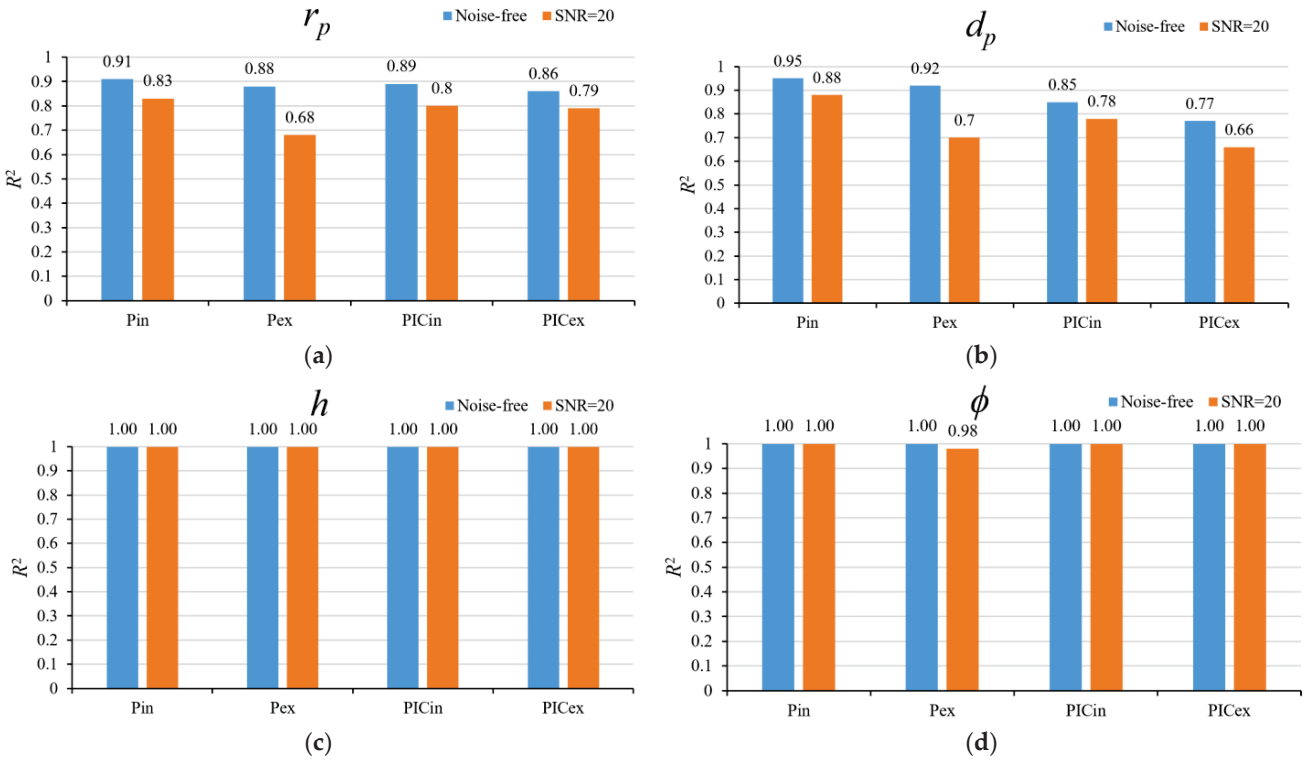


Figure 13. Comparison of R^2 between noise-free and SNR = 20 scenarios for different corrosion types.

Finally, it is important to point out that the CNN classification and regression models developed in this study are specifically tuned to certain pipeline and MFL tool attributes, such as the pipe wall thickness, magnet properties, lift-off values, and defect shapes. Therefore, further investigations are needed to determine how well the CNN model performs with various pipe wall thicknesses and different MFL tool parameters. Further studies are also needed to take into account the movement of the MFL tool into the prediction models.

5. Conclusions

This paper reports a novel study that focuses on the classification and sizing of pinhole corrosions on pipelines using deep learning algorithms. We propose a CNN classification model to classify six different types of corrosions (i.e., G_{in}, G_{ex}, P_{in}, P_{ex}, PIC_{in}, and PIC_{ex}) on pipelines and a CNN regression model to estimate the sizes and location of the pinhole defects based on MFL signals generated using magneto-static FEA. Extensive 3D parametric FEA cases involving box-shaped general corrosions and cylinder-shaped pinholes are used to simulate the 3D MFL signals by varying the defect depth, length, width, and longitudinal and circumferential locations. The CNN classification and regression models are then trained and validated using the simulated MFL signals.

The proposed classification model is shown to have excellent accuracy in classifying six types of corrosion defects with a misprediction rate of 0.34% (33/9600 = 0.34%). The

proposed regression model is highly accurate in predicting the location of the pinhole and achieves good accuracy in predicting the depth and diameter of the pinhole. Through a comparative analysis of the CNN regression model with noise-free and noisy signals (SNR = 20) as input, it is observed that the noise impact on the predictive accuracy of the regression model is moderate. This study demonstrates the application of deep learning algorithms to facilitate the integrity management of pipelines containing complex-shaped corrosion defects.

Author Contributions: Y.S.: Investigation, Visualization, Formal analysis, Methodology, Writing—original draft. W.Z.: Funding acquisition, Supervision, Conceptualization, Methodology, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the financial support received from the Natural Sciences and Engineering Research Council of Canada (NSERC) (Grant No. RGPIN-2019–05160) and the Faculty of Engineering of the University of Western Ontario.

Data Availability Statement: Data are available on request from the authors.

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

References

1. Zhou, Q.; Wu, W.; Liu, D.; Li, K.; Qiao, Q. Estimation of corrosion failure likelihood of oil and gas pipeline based on fuzzy logic approach. *Eng. Fail. Anal.* **2016**, *70*, 48–55. [CrossRef]
2. Murphy, J.F. Nightmare pipeline failures, fantasy planning, black swans, and integrity management—A review. *Process Saf. Prog.* **2015**, *34*, 207. [CrossRef]
3. Shen, Y.; Zhou, W. A comparison of onshore oil and gas transmission pipeline incident statistics in Canada and the United States. *Int. J. Crit. Infrastruct. Prot.* **2024**, *45*, 100679. [CrossRef]
4. Vanaei, H.R.; Eslami, A.; Egbewande, A. A review on pipeline corrosion, in-line inspection (ILI), and corrosion growth rate models. *Int. J. Press. Vessels Pip.* **2017**, *149*, 43–54. [CrossRef]
5. Song, H.; Yang, L.; Liu, G.; Tian, G.; Ona, D.I.; Song, Y.; Li, S. Comparative analysis of in-line inspection equipments and technologies. *IOP Conf. Ser. Mater. Sci. Eng.* **2018**, *382*, 032021. [CrossRef]
6. Pipeline Operators Forum. Specifications and Requirements for In-Line Inspection of Pipelines. Version 2021. Available online: https://pipelineoperators.org/documents (accessed on 18 May 2022).
7. Sutherland, J.; Bluck, M.; Pearce, J.; Quick, E. Validation of latest generation MFL in-line inspection technology leads to improved detection and sizing specification for pinholes, pitting, axial grooving and axial slotting. In Proceedings of the ASME 2010 8th International Pipeline Conference, Calgary, AB, Canada, 27 September–1 October 2010.
8. Peng, X.; Anyaoha, U.; Liu, Z.; Tsukada, K. Analysis of magnetic-flux leakage (MFL) data for pipeline corrosion assessment. *IEEE Trans. Magn.* **2020**, *56*, 1–15. [CrossRef]
9. Han, W.; Yang, P.; Xia, F.; Xue, Y. Magnetic flux leakage signal inversion of corrosive flaws based on modified genetic local search algorithm. *J. Shanghai Jiaotong Univ. (Sci.)* **2009**, *14*, 168–172. [CrossRef]
10. Priewald, R.H.; Magele, C.; Ledger, P.D.; Pearson, N.R.; Mason, J.S.D. Fast magnetic flux leakage signal inversion for the reconstruction of arbitrary defect profiles in steel using finite elements. *IEEE Trans. Magn.* **2012**, *49*, 506–516. [CrossRef]
11. Hwang, K.; Mandayam, S.; Udpa, S.S. Characterization of gas pipeline inspection signals using wavelet basis function neural networks. *NDT E Int.* **2000**, *33*, 531–545. [CrossRef]
12. Ramuhalli, P.; Udpa, L.; Udpa, S.S. Neural network based inversion algorithms in magnetic flux leakage nondestructive evaluation. *J. Appl. Phys.* **2003**, *93*, 82748276. [CrossRef]
13. Han, W.H.; Que, P.W. 2-D defect reconstruction from MFL signals based on genetic optimization algorithm. In Proceedings of the IEEE 2005 International Conference on Industrial Technology, Hong Kong, China, 14–17 December 2005.
14. Kandroodi, M.R.; Araabi, B.N.; Bassiri, M.M.; Ahmadabadi, M.N. Estimation of depth and length of defects from magnetic flux leakage measurements: Verification with simulations, experiments, and pigging data. *IEEE Trans. Magn.* **2016**, *53*, 1–10. [CrossRef]
15. Feng, J.; Li, F.; Lu, S.; Liu, J. Fast reconstruction of defect profiles from magnetic flux leakage measurements using a RBFNN based error adjustment methodology. *IET Sci. Meas. Technol.* **2017**, *11*, 262–269. [CrossRef]
16. Yao, G.; Lei, T.; Zhong, J. A review of convolutional-neural-network-based action recognition. *Pattern Recognit. Lett.* **2019**, *118*, 14–22. [CrossRef]
17. Taye, M.M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation* **2023**, *11*, 52. [CrossRef]
18. Lu, S.; Feng, J.; Zhang, H.; Liu, J.; Wu, Z. An estimation method of defect size from MFL image using visual transformation convolutional neural network. *IEEE Trans. Ind. Inform.* **2018**, *15*, 213–224. [CrossRef]

19. Shen, Y.; Zhou, W. Development of a convolutional neural network model to predict the size and location of corrosion defects on pipelines based on magnetic flux leakage signals. *Int. J. Press. Vessels Pip.* **2023**, *207*, 105–123. [CrossRef]
20. Wang, H.A.; Chen, G. Defect size estimation method for magnetic flux leakage signals using convolutional neural networks. *Insight* **2020**, *62*, 86–91. [CrossRef]
21. EGIG. *10th Report of the European Gas Pipeline Incident Data Group (Period 1970–2016)*; EGIG: Groningen, The Netherlands, 2018; Doc. number VA 17.R.0395.
22. Zhang, H.; Sha, S.; Willis, C.; Qingshan, F.; Chen, P. Feasibility study of pinhole inspection via magnetic flux leakage and hydrostatic testing in oil & gas pipelines. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1043*, 022053.
23. Feng, Q.; Yan, B.; Chen, P.; Shirazi, S.A. Failure analysis and simulation model of pinhole corrosion of the refined oil pipeline. *Eng. Fail. Anal.* **2019**, *106*, 1–28. [CrossRef]
24. Subramanian, C. Localized pitting corrosion of API 5L grade A pipe used in industrial fire water piping applications. *Eng. Fail. Anal.* **2018**, *92*, 405–417. [CrossRef]
25. Askari, M.; Aliofkhazraei, M.; Afroukhteh, S. A comprehensive review on internal corrosion and cracking of oil and gas pipelines. *J. Nat. Gas Sci. Eng.* **2019**, *71*, 102971. [CrossRef]
26. Kadhim, K.N.; Al-Rufaye, A.H.R. The effects of uniform transverse magnetic field on local flow and velocity profile. *Int. J. Civ. Eng. Technol.* **2016**, *7*, 140–151.
27. Ji, F.; Wang, C.; Sun, S.; Wang, W. Application of 3-D FEM in the simulation analysis for MFL signals. *Insight* **2009**, *51*, 32–35. [CrossRef]
28. Bubenik, T. *Electromagnetic Methods for Detecting Corrosion in Underground Pipelines: Magnetic Flux Leakage (MFL)*; Underground Pipeline Corrosion; Woodhead Publishing: Cambridgeshire, UK, 2014; pp. 215–226.
29. Shi, Y.; Zhang, C.; Li, R.; Cai, M.; Jia, G. Theory and application of magnetic flux leakage pipeline detection. *Sensors* **2015**, *15*, 31036–31055. [CrossRef]
30. Zhang, Y.; Ye, Z.; Wang, C. A fast method for rectangular crack sizes reconstruction in magnetic flux leakage testing. *NDT E Int.* **2009**, *42*, 369–375. [CrossRef]
31. Walker, J. *In-Line Inspection of Pipelines: Advanced Technologies for Economic and Safe Operation of Oil and Gas Pipelines*; Verlag Moderne Industrie: Landsberg am Lech, Germany, 2010.
32. Liu, Y.; Gao, X.; Wang, Y.; Yang, X. Sensitive parameters' optimization of the permanent magnet supporting mechanism. *J. Mech. Sci. Technol.* **2014**, *28*, 2707–2714. [CrossRef]
33. Yang, L.; Zhang, G.; Liu, G.; Gao, S. Effect of lift-off on pipeline magnetic flux leakage inspection. In Proceedings of the 2008 17th World Conference on Nondestructive Testing, Shanghai, China, 25–28 October 2008.
34. Ireland, R.C.; Torres, C.R. Finite element modelling of a circumferential magnetizer. *Sens. Actuators A Phys.* **2006**, *129*, 197–202. [CrossRef]
35. Azizzadeh, T.; Safizadeh, M.S. Three-dimensional finite element and experimental simulation of magnetic flux leakage-type NDT for detection of pitting corrosions. In Proceedings of the 2017 4th Iranian International NDT Conference (IRNDT), Tehran, Iran, 21–22 February 2017.
36. Chen, L.; Li, X.; Qin, G.; Lu, Q. Signal processing of magnetic flux leakage surface flaw inspect in pipeline steel. *Russ. J. Nondestruct. Test.* **2008**, *44*, 859–867. [CrossRef]
37. Piao, G.; Guo, J.; Hu, T.; Leung, H. The effect of motion-induced eddy current on high-speed magnetic flux leakage (MFL) inspection for thick-wall steel pipe. *Res. Nondestruct. Eval.* **2020**, *31*, 48–67. [CrossRef]
38. Li, F.; Feng, J.; Zhang, H.; Liu, J.; Lu, S.; Ma, D. Quick reconstruction of arbitrary pipeline defect profiles from MFL measurements employing modified harmony search algorithm. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 9. [CrossRef]
39. Fan, X.; Dai, M.; Liu, C.; Wu, F.; Yan, X.; Feng, Y.; Feng, Y.; Su, B. Effect of image noise on the classification of skin lesions using deep convolutional neural networks. *Tsinghua Sci. Technol.* **2019**, *25*, 425–434. [CrossRef]
40. Acharya, U.R.; Oh, S.L.; Hagiwara, Y.; Tan, J.H.; Adam, M.; Gertych, A.; San Tan, R. A deep convolutional neural network model to classify heartbeats. *Comput. Boil. Med.* **2017**, *89*, 389–396. [CrossRef] [PubMed]
41. Tesfai, H.; Saleh, H.; Al-Qutayri, M.; Mohammad, M.B.; Tekeste, T.; Khandoker, A.; Mohammad, B. Lightweight shufflenet based cnn for arrhythmia classification. *IEEE Access.* **2022**, *10*, 111842–111854. [CrossRef]
42. West, N.E.; O'shea, T. Deep architectures for modulation recognition. In Proceedings of the 2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Baltimore, MD, USA, 6–9 March 2017.
43. Baranwal, S.; Khandelwal, S.; Arora, A. Deep learning convolutional neural network for apple leaves disease detection. In Proceedings of the 2019 International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur, India, 26–28 February 2019.
44. Virupakshappa, K.; Marino, M.; Oruklu, E. A multi-resolution convolutional neural network architecture for ultrasonic flaw detection. In Proceedings of the 2018 IEEE International Ultrasonics Symposium (IUS), Kobe, Japan, 22–25 October 2018.
45. Bilmes, J. Underfitting and Overfitting in Machine Learning. UW ECE Course Notes. 2020. Available online: https://people.ece.uw.edu/bilmes/classes/ee511/ee511_spring_2020/overfitting_underfitting.pdf (accessed on 20 March 2023).
46. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 2015 International Conference on Machine Learning, Lille, France, 6–11 July 2015.

47. Thakkar, V.; Tewary, S.; Chakraborty, C. Batch normalization in convolutional neural networks—A comparative study with CIFAR-10 data. In Proceedings of the IEEE 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), IIEST Shibpur, West Bengal, India, 12–13 January 2018.
48. Bjorck, N.; Gomes, C.P.; Selman, B.; Weinberger, K.Q. Understanding batch normalization. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 1–12.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

algorithms

MDPI

Review

UAV (Unmanned Aerial Vehicle): Diverse Applications of UAV Datasets in Segmentation, Classification, Detection, and Tracking

Md. Mahfuzur Rahman [1], Sunzida Siddique [2,*], Marufa Kamal [3], Rakib Hossain Rifat [4] and Kishor Datta Gupta [5]

[1] Silicon Orchard Research and Analytics Lab, Dhaka 1216, Bangladesh; mahim@siliconorchard.com
[2] Department of Computer Science and Engineering, Daffodil International University, Dhaka 1216, Bangladesh
[3] Department of Computer Science and Engineering, BRAC University, Dhaka 1212, Bangladesh; marufa.kamal1@g.bracu.ac.bd
[4] Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA; rrifat@ttu.edu
[5] Department of Cyber Physical Systems, Clark Atlanta University, Atlanta, GA 30314, USA; kgupta@cau.edu
* Correspondence: sunzida15-9667@diu.edu.bd

Abstract: Unmanned Aerial Vehicles (UAVs) have transformed the process of data collection and analysis in a variety of research disciplines, delivering unparalleled adaptability and efficacy. This paper presents a thorough examination of UAV datasets, emphasizing their wide range of applications and progress. UAV datasets consist of various types of data, such as satellite imagery, images captured by drones, and videos. These datasets can be categorized as either unimodal or multimodal, offering a wide range of detailed and comprehensive information. These datasets play a crucial role in disaster damage assessment, aerial surveillance, object recognition, and tracking. They facilitate the development of sophisticated models for tasks like semantic segmentation, pose estimation, vehicle re-identification, and gesture recognition. By leveraging UAV datasets, researchers can significantly enhance the capabilities of computer vision models, thereby advancing technology and improving our understanding of complex, dynamic environments from an aerial perspective. This review aims to encapsulate the multifaceted utility of UAV datasets, emphasizing their pivotal role in driving innovation and practical applications in multiple domains.

Keywords: UAV (unmanned aerial vehicle); UAV datasets; object detection; semantic segmentation; action recognition; event recognition; aerial; surveillance

1. Introduction

Unmanned Aerial Vehicles (UAVs) [1], commonly referred to as drones, have revolutionized the way we collect and analyze data from above, offering unparalleled versatility and efficiency across various research fields. This review examines the wide-ranging applications of UAV datasets, highlighting their significant advancements and potential across multiple domains. UAV datasets include various data types, such as satellite imagery, drone-captured images, and videos. Datasets may be classified as unimodal, focusing on a single data type, or multimodal, which incorporates multiple data types to provide deeper and more comprehensive insights.

UAV datasets facilitate the assessment of disaster damage by enabling the classification of damage from natural disasters through advanced semantic segmentation and annotation techniques. Training computer vision models with these datasets enables the automation of aerial scene classification for disaster events, thereby improving response and recovery efforts. The extraction of information and detection of objects from UAV-captured data is essential for tasks such as action recognition, which involves analyzing human behavior from aerial imagery, including the recognition of aerial gestures and the classification of disaster events.

A significant application of UAV datasets is in aerial surveillance, which facilitates advanced research by the convergence of computer vision, robotics, and surveillance.

The datasets facilitate event recognition in aerial videos, contributing to the monitoring of urban environments and traffic systems. The application of pre-trained models and transfer learning techniques enhances the effectiveness of UAV datasets, facilitating the swift implementation of advanced models for event recognition and tracking.

UAV datasets improve object recognition in urban surveillance by offering extensive top-down and side views. This enables tasks including categorization, verification, object detection, and tracking of individuals and vehicles. UAV datasets play a critical role in the understanding and management of forest ecosystems by facilitating the segmentation of individual trees, an essential aspect of sustainable forest management.

The versatility of UAV datasets encompasses multiple domains, such as the development of speech recognition systems for UAV control through video capture and object tracking in low-light conditions, which is crucial for nighttime surveillance operations. Innovative UAV designs, including bionic drones with flapping wings, have resulted in specialized video datasets utilized for single object tracking (SOT) [2]. This illustrates the extensive scope and potential of UAV datasets in improving real-time object tracking across diverse lighting conditions.

UAV datasets serve as a fundamental resource for advanced research and practical applications in various fields. This review examines the specific applications and advantages of these datasets, emphasizing their contribution to technological advancement and enhancing our comprehension of complex, dynamic environments from an aerial viewpoint.

The subsequent sections provide a comprehensive exposition of the contributions made by our study, which can be stated as follows:

- Our study is driven by the increasing importance of UAV datasets in several research domains such as object detection, traffic monitoring, action identification, surveillance in low-light conditions, single object tracking, and forest segmentation utilizing point cloud or LiDAR point process modeling. Through an in-depth analysis of current datasets, their uses, and prospects, this paper intends to provide valuable insights that will assist researchers in harnessing these resources for creative solutions. Furthermore, they will acquire knowledge of existing constraints and prospective opportunities, enhancing their research endeavors.
- We conduct an extensive analysis of a dataset consisting of 15 UAVs, showcasing its diverse applications in research.
- We emphasize the applications and advancements of several novel methods utilizing these datasets based on UAVs.
- Our study also delves into the potential for future research and the feasibility of utilizing these UAV datasets, engaging in in-depth discussions on these topics.

2. Methodology

UAVs encompass a diverse range of applications, requiring a thorough investigation to examine and define the extensive utilization of UAV datasets. We aimed to comprehend how these datasets can be employed in different research and project scenarios. To accomplish this, we implemented an exhaustive search for UAV datasets, initially narrowing our focus to the keyword "satellite or drone image datasets". The initial search led to the identification of "UAV datasets". After acknowledging the potential of UAV datasets, we conducted further research in this field, identifying their diverse applications in object detection, tracking, and event detection, as well as semantic segmentation and single object tracking.

To gather relevant UAV datasets, we conducted systematic searches on the Internet, employing a range of keywords and search terms related to UAVs and their applications. We specifically looked for datasets that showed off the adaptability of UAVs, choosing those that researchers had proposed and used in other research contexts. This approach ensured that the datasets we included were novel and provided diverse examples of UAV applications.

We identified and collected 15 UAV image datasets for inclusion in our study. Our selection criteria focused on datasets that showcased a variety of use cases, including traffic systems (car identification, person identification, and surveillance systems), damage classification from disasters, and other object detection and segmentation tasks. Each dataset was thoroughly reviewed and analyzed to understand its characteristics, intended use, and underlying methodologies.

Our analysis involved a detailed examination of the datasets, resulting in the comprehensive report included in this paper. This report outlines the behavior, agenda, and applications of each dataset, providing insights into their respective fields of use. By presenting these findings, we aim to highlight the versatility and potential of UAV datasets in advancing various research domains. Figure 1 depicts the sequential process of our work.

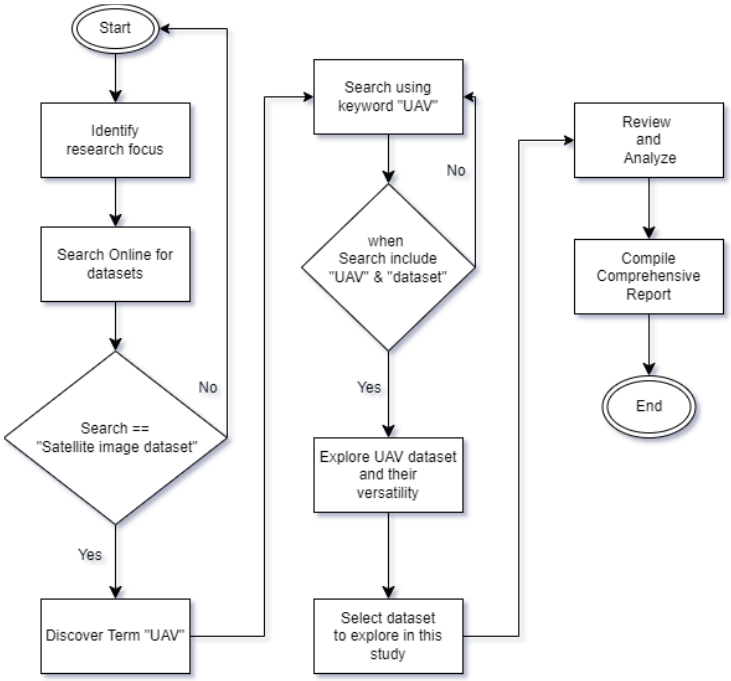


Figure 1. Workflow of this study.

Search Terms

We obtained the datasets surveyed in this paper mostly from the following website: https://paperswithcode.com/ (accessed on 12 November 2024). Before we found this website, we used various search terms to search for UAV datasets and came across the website through the search process.

Example search strings:

- (“unmanned aerial vehicle” OR UAV OR drone OR Satellite) AND (“dataset” OR “image dataset” OR “dataset papers”)
- (UAV OR “unmanned aerial vehicle”) AND (“disaster dataset” OR “traffic surveillance”)

These search strings and keywords facilitated a broad yet focused search, enabling us to gather a diverse set of UAV datasets that demonstrate their wide-ranging applications and research potential.

3. Literature Review

UAVs function without a human pilot present and can be remotely commanded or autonomously managed by computers onboard. They are used in diverse fields such as surveillance, aerial photography, agriculture, environmental monitoring, and military operations. UAVs greatly enhance computer vision jobs by collecting several forms of image and video data, in addition to satellite images, within the realm of datasets.

In addition to the datasets examined in this work, recent research has investigated novel applications of UAVs beyond conventional purposes. These include real-time 3D AQI mapping for air quality assessment [3], wildfire detection systems that integrate UAVs with satellite and ground sensors [4], federated learning frameworks for energy-efficient UAV swarm operations [5], and extensive surveys that tackle air pollution monitoring difficulties. These efforts underscore the adaptability of UAVs in tackling urgent environmental and computational issues, while also revealing prospects for additional research in particular fields like disaster management and machine learning integration.

Tables 1 and 2 encapsulate the datasets examined, which form the foundation of our analysis.

Table 1. Summary of research and findings of UAV datasets discussed.

Dataset	Dataset Details	Paper Findings	Limitations	Future Work
RescueNet [6]	High-resolution images with pixel-level annotations for 10 classes, collected via UAVs after Hurricane Michael.	Attention-based and transformer-based methods performed best. Transfer learning from RescueNet to FloodNet improved segmentation.	Time-consuming annotation process and potential lack of comprehensive post-disaster elements.	Further evaluation across different disaster scenarios to enhance robustness.
UAV-Human [7]	Comprises 67,428 multi-modal video sequences for action recognition, pose estimation, person re-identification, and attribute recognition.	Highest action recognition accuracies with night-vision and IR videos. Pose estimation methods achieved mAP scores of 56.5 and 56.9.	Potential overfitting, lack of subject diversity, and constrained capturing conditions.	Increase sample size and diversity, and capture conditions to enhance model robustness and generalization.
AIDER [8]	Includes 2565 manually gathered images of disaster events with augmentations.	Development of ERNet, achieving near state-of-the-art accuracy (90%) and over 50 fps on a CPU platform.	Does not extensively discuss real-time implementation challenges, robustness in diverse conditions, or hyperparameter tuning.	Integrate ERNet with algorithms for detecting people and vehicles, use additional modalities like infrared cameras, and optimize the model for improved generalization and accuracy.
AU-AIR [9]	A total 32,823 labeled video frames with object annotations and flight data.	YOLOv3-tiny and MobileNetv2-SSD Lite for real-time object detection on UAVs showed potential for onboard computer applicability.	Focus on traffic surveillance may limit applicability to other scenarios, lacks advanced baselines for tasks like UAV navigation.	Enhance dataset diversity, incorporate more environmental contexts, and develop additional baselines leveraging sensor data for broader applications.
ERA [10]	A total 2,864 videos capturing events in a wide range of settings and sizes.	DenseNet-201 achieved the highest accuracy of 62.3% in single-frame classification.	Dataset size, class imbalance, and challenge of distinguishing events from normal videos.	Focus on attribute recognition, temporal cue exploitation, and addressing challenging cases like human action recognition.

Table 1. *Cont.*

Dataset	Dataset Details	Paper Findings	Limitations	Future Work
UAVid [11]	Contains 30 video sequences featuring high-resolution 4K images with 8 labeled classes for semantic segmentation.	Multi-Scale-Dilation Net achieved an average IoU score of around 50%.	Class imbalance, particularly in urban street scenes, potentially affecting model performance and generalization.	Balance method complexity with practical implementation, expand dataset size and object categories, address class imbalance, and explore other applications like object detection and tracking.

Table 2. Summary of research and findings of UAV datasets discussed.

Dataset	Dataset Details	Paper Findings	Limitations	Future Work
VRAI [12]	A total 137,613 images of 13,022 vehicles with detailed annotations captured by two UAVs.	Outperforms existing methods in vehicle ReID techniques using GANs and attention models.	Comparison scope, domain specificity, annotation complexity, scalability, and real-world deployment insights.	Explore transfer learning, enhance scalability, integrate advanced techniques, focus on real-world applications, and improve annotation strategies.
FOR-Instance [13]	Five collections from around the world for individual tree segmentation from UAV-based laser scanning data.	Supports both instance and semantic segmentation, adaptable to deep learning frameworks.	Potential overfitting, lack of generalizability to other forest types, challenges with unclassified points.	Incorporate more data types, develop advanced deep learning architectures, study tree species classification, and conduct longitudinal studies on forest changes.
VERI-Wild [14]	Over 400,000 images of 40,671 vehicle IDs captured from a real CCTV camera system over one month.	FDA-Net outperforms existing methods, achieving highest Rank-1 and Rank-5 accuracies.	Potential biases due to urban district focus and dataset-specific adversarial scheme.	Explore more challenging real-world factors, generate comprehensive datasets, and leverage GANs to improve cross-view ReID performance.
UAV-Assistant [15]	Data synthesis pipeline combining egocentric UAV views and exocentric user views with smooth silhouette loss.	Smooth silhouette loss enhances 3D pose estimation accuracy.	Lack of real-world data poses a challenge to generalizability, and determining optimal kernel size for smoothing filter.	Optimize parameters, explore additional loss functions, and validate approach in real-world scenarios.
KITE [16]	Focus on UAV control speech recognition with multimodal systems.	Recurrent neural networks (RNNs) for language modeling and visual cues integration.	Imperfect command–image associations, biases from semi-automatic methods for training data generation.	Address biases, enhance dataset generalizability, and explore other architectural decisions.
UAV-Gesture [17]	Contains 119 high-definition video clips of 13 gestures for UAV navigation and command.	Annotates body joints and gesture classes in 37,151 frames using an extended version of VATIC.	Limited gesture set and non-expert actors may affect dataset quality.	Leverage dataset for gesture and action recognition in UAV control, expand and refine dataset for broader research applications.

Table 2. *Cont.*

Dataset	Dataset Details	Paper Findings	Limitations	Future Work
UAVDark 135 [18]	Over 125k manually annotated frames for dark tracking methods.	ADTrack demonstrates superiority in bright and dark conditions.	Lacks broader comparison with other state-of-the-art trackers.	Further research on real-time tracking algorithms, new image enhancement methods, multi-sensor fusion techniques, and hardware optimization strategies.
DarkTrack2021 [19]	Has 110 annotated sequences totaling over 100,000 frames for low-light UAV tracking.	SCT demonstrated significant performance gains for nighttime UAV tracking.	Comparisons with daytime tracking scenarios needed to be improved.	Explore advanced transformer architectures, attention mechanisms, noise reduction strategies, and real-world validation.
BioDrone [20]	Comprises 600 videos annotated and labeled at the frame level for single object tracking using bionic drone-based systems.	Comprehensive evaluation platform for robust vision research.	Focus on bionic UAVs may limit generalization, potential biases in annotations.	Improve tracking algorithms, address computational complexity and real-time performance.

3.1. RescueNet

Maryam Rahnemoonfar, Tashnim Chowdhury, and Robin Murphy presented the RescueNet [6] dataset in their paper, which focuses on post-disaster scene understanding using UAV imagery. The dataset contains high-resolution images with detailed pixel-level annotations for ten classes of objects, including buildings, roads, pools, and trees, which were collected by UAVs following Hurricane Michael. The authors employed state-of-the-art segmentation models like Attention UNet [21], PSPNet [22], and DeepLabv3 [23], achieving superior performance with attention-based and transformer-based methods. The findings demonstrated RescueNet's effectiveness in improving damage assessment and response strategies, with transfer learning outperforming other datasets like FloodNet [24]. The dataset was observed to have limited generalization to other domains and to require a time-consuming annotation process, despite its detailed annotations.

3.2. UAV-Human

Tianjiao Li et al. developed the UAV-Human [7] dataset, a comprehensive benchmark for improving human behavior understanding with UAVs. The dataset contains 67,428 multi-modal video sequences with 119 subjects for action recognition, 22,476 frames for pose estimation, 41,290 frames for person re-identification with 1144 identities, and 22,263 frames for attribute recognition, all captured over three months in various urban and rural locations under varying conditions. The data encompass RGB videos, depth maps, infrared sequences, and skeleton data. The authors used methods such as HigherHR-Net [25], AlphaPose [26], and the Guided Transformer I3D framework to recognize actions while addressing fisheye video distortions [27,28] and leveraging multiple data modalities. The results demonstrated the dataset's effectiveness in improving action recognition, pose estimation, and re-identification tasks, with models showing significant performance improvements. The UAV-Human dataset stands out as a reliable benchmark, encouraging the creation of more effective UAV-based human behavior analysis algorithms.

3.3. AIDER

Christos Kyrkou and Theocharis Theocharides introduced the AIDER [8] dataset, which is intended for disaster event classification using UAV aerial images. The dataset

contains 2565 images of Fire/Smoke, Flood, Collapsed Building/Rubble, Traffic Accidents, and Normal cases, which were manually collected from various sources—mainly from UAVs. To increase variability and combat overfitting, images were randomly augmented with rotations, translations, and color shifting. The paper presents ERNet, a lightweight CNN designed for efficient classification on embedded UAV platforms. ERNet, which uses components from architectures such as VGG16 [29], ResNet [30], and MobileNet [31], incorporates early downsampling to reduce computational costs. When tested on both embedded platforms attached to UAVs and desktop CPUs, ERNet achieved almost perfect accuracy (90%) while running three times faster on embedded platforms. This showed that it is a good choice for real-time applications that do not need a lot of memory. The study emphasizes the benefits of combining ERNet with other detection algorithms to improve situational awareness in emergency response.

3.4. AU-AIR

In their paper, Ilker Bozcan and Erdal Kayacan presented the AU-AIR [9] dataset, a comprehensive UAV dataset designed for traffic surveillance. The dataset comprises 32,823 labeled video frames with annotations for eight traffic-related object categories, along with multi-modal data including GPS coordinates, altitude, IMU data [32], and velocity. To establish a baseline for real-time performance in UAV applications, the authors train and evaluate two mobile object detectors on this dataset: YOLOv3-Tiny [33] and MobileNetv2-SSDLite [34]. The findings highlight the difficulties of object detection in aerial images, emphasizing the importance of datasets tailored to mobile detectors. The study highlights the dataset's potential for furthering research in computer vision, robotics, and aerial surveillance, while also acknowledging limitations and suggesting future improvements for broader applicability.

3.5. ERA

Lichao Mou et al. introduced the ERA [10] dataset, a comprehensive collection of 2864 labeled video snippets for twenty-four event classes and one normal class, designed for event recognition in UAV videos. The videos, sourced from YouTube, are 5 s long, 640×640 pixels, and run at 24 fps, ensuring a diverse dataset that includes both high-quality and extreme condition footage. The paper employs various deep learning models, including VGG-16, ResNet-50, DenseNet-201 [35], and video classification models like I3D-Inception-v1, to benchmark event recognition. DenseNet-201 achieved the highest performance with an overall accuracy of 62.3% in single-frame classification. The findings highlight the difficulties of recognizing events in a variety of environments and scales, noting that while models can identify specific events such as traffic congestion and smoke, they struggle with conditions such as night and snow scenes, indicating the need for improved attribute recognition and temporal cue exploitation in future research.

3.6. UAVid

Ye Lyu et al. introduced the UAVid [11] dataset in their paper, which addresses the need for semantic segmentation in urban scenes from the perspective of UAVs. The UAVid dataset consists of 30 video sequences with 4K high-resolution images, which capture top and side views for improved object recognition and include eight labeled classes. The paper highlights the challenges of large-scale variation, moving object recognition, and temporal consistency. The effectiveness of deep learning techniques, such as Multi-Scale-Dilation net, which is a novel technique proposed by the author, was evaluated and resulted in an average Intersection over Union [36] (IoU) score of approximately 50%. Further enhancements were observed by employing spatial–temporal regularization methods like FSO [37] and 3D CRF [38]. The dataset's applicability extends to traffic monitoring, population density analysis, and urban greenery monitoring, showcasing its potential for diverse urban surveillance applications. The paper also discusses the dataset's

class imbalance and suggests future expansions and optimizations to enhance its utility for semantic segmentation and other UAV-based tasks.

3.7. VRAI

Peng Wang et al. introduced the VRAI [12] dataset, which is the largest vehicle re-identification (ReID) dataset containing over 137,613 images of 13,022 vehicles. This UAV-based dataset includes annotations for unique IDs, color, vehicle type, attributes, and distinguishing features, capturing a wide range of view angles and poses from UAVs flying between 15 m and 80 m. The study devised an innovative vehicle ReID algorithm that utilizes weight matrices, weighted pooling, and comprehensive annotations to identify distinctive components. This algorithm surpasses both the baseline and the most advanced techniques currently available. The paper utilizes a comprehensive strategy to perform vehicle ReID using aerial images, showcasing its effectiveness through a range of experiments. Ablation study results demonstrate that the novel Multi-task + DP model, which integrates attribute classification and additional triplet loss on weighted features, exhibits superior performance compared to less complex models. The proposed method outperforms ground-based methods such as MGN [39], RNN-HA [40], and RAM [41], because it can easily handle different view angles in UAV images. Weighted feature aggregation improves performance, as evidenced by the enhanced mean average precision (mAP) and cumulative match characteristic (CMC) metrics. Human performance evaluation highlights the algorithm's strength in fine-grained recognition, though humans still excel in detailed tasks. The study suggests further research to improve flexibility, scalability, and real-world application of the algorithm.

3.8. FOR-Instance

For semantic and instance segmentation of individual trees, Stefano Puliti et al. presented the FOR-Instance [13] dataset in their paper "FOR-Instance: a UAV laser scanning benchmark dataset for semantic and instance segmentation of individual trees". This dataset addresses a deficiency in the market for machine learning-ready datasets and standardized benchmarking infrastructure by providing publicly accessible annotated forest data for point cloud segmentation tasks. The major objective is to utilize data from unmanned aerial vehicle (UAV) laser scanning to accurately identify and distinguish individual trees. The dataset has comprehensive annotations utilized for training and evaluation, consisting of five meticulously selected samples from various forest types globally. In deep learning, the dataset is partitioned into several subsets for training and validation purposes. In picture segmentation research, rasterized canopy height models are employed alongside either raw point clouds or two-dimensional projections. The FOR-Instance dataset was beneficial for the examination and evaluation of sophisticated segmentation techniques. This underscores the need to understand forest ecosystems and develop sustainable management strategies. The standardization of the dataset in 3D forest scene segmentation study mitigates existing methodological shortcomings, including overfitting and insufficient comparability.

3.9. VERI-Wild

Yihang Lou et al. presented the VERI-Wild [14] dataset, the largest vehicle ReID dataset to date, in their paper. Over 400,000 photos of 40,000 vehicle IDs are included in the dataset, which was collected over the course of a month in an urban district using 174 CCTV cameras. The dataset poses a formidable challenge for ReID algorithms due to its inclusion of diverse conditions such as varying backgrounds, lighting, obstructions, perspectives, weather, and vehicle types. The authors introduced FDA-Net, a novel technique for vehicle ReID, to enhance the model's ability to distinguish between different vehicles. FDA-Net combines a feature distance adversary network with a hard negative generator and embedding discriminator. After being tested on the VERI-Wild dataset and other established datasets, FDA-Net surpassed various standard methods, achieving higher

accuracies in Rank-1 and Rank-5. This demonstrates the effectiveness of FDA-Net in vehicle ReID tasks. The method's ability to generate hard negatives significantly improved model performance, highlighting its potential for advancing vehicle ReID research in real-world scenarios.

3.10. UAV-Assistant

G. Albanis and N. Zioulis et al. introduced the UAV-Assistant [15] (UAVA) dataset in their paper. The dataset was created using a data synthesis pipeline to generate realistic multimodal data, including exocentric and egocentric views from UAVs. The dataset is suitable for training a model aimed at estimating an individual's pose by integrating a new smooth silhouette loss alongside a direct regression objective. The dataset serves as a valuable resource for training a model capable of precisely identifying a person's position by integrating a distinctive smooth silhouette loss with a direct regression objective. It also uses differentiable rendering techniques to help the model learn from both real and fake data. The study highlights the critical role of tuning the kernel size for the smoothing filter to optimize model performance. The suggested smooth silhouette loss surpasses conventional silhouette loss functions by reducing discrepancies and enhancing the accuracy of 3D pose estimation. This approach specifically tackles the lack of available data for estimating the three-dimensional position and orientation of unmanned aerial vehicles (UAVs) in non-hostile environments. It is different from existing datasets that primarily focus on remote sensing or drones with malicious intent. The paper underscores the need for further research on rendering techniques, parameter optimization, and real-world validations to enhance the model's generalizability and robustness.

3.11. KITE

The KITE [16] dataset, created to improve speech recognition systems for UAV control, was presented by Dan Oneata and Horia Cucu in their paper. The KITE eval dataset is a comprehensive collection that includes 2880 spoken commands, alongside corresponding audio and images. It is specifically designed for UAV operations and covers a range of commands related to movement, camera usage, and specific scenarios. The authors employed time-delay neural networks [42] (which are implemented in Kaldi [43]) and recurrent neural networks to perform language modeling. They initialized the models with out-of-domain datasets and subsequently fine-tuned them for UAV tasks. The study emphasizes the efficacy of customizing language models for UAV-specific instructions, showcasing substantial enhancements in speech recognition precision through domain adaptation. Future directions include grounding uttered commands in images for enhanced context understanding and improving the acoustic model's robustness to outdoor noises.

3.12. UAV-Gesture

A. Perera et al. introduced the UAV-Gesture [17] dataset, which addresses the lack of research on gesture-based UAV control in outdoor settings. This dataset aims to fill the existing research gap, as most studies in this field are focused on indoor environments. The dataset consists of 119 high-definition video clips, totaling 37,151 frames, captured in an outdoor setting using a 3DR Solo UAV and a GoPro Hero 4 Black camera. The dataset comprises annotations of 13 body joints and gesture classes for all frames, encompassing gestures appropriate for UAV navigation and command. The dataset was captured with variations in phase, orientation, and camera movement to augment realism. The authors employed an extended version of the VATIC [44] tool for annotation and utilized a Pose-based Convolutional Neural Network [45] (P-CNN) for gesture recognition. This approach resulted in a baseline accuracy of 91.9%. This dataset facilitates extensive research in gesture recognition, action recognition, human pose recognition, and UAV control, showcasing its efficacy and potential for real-world applications.

3.13. UAVDark135

In their research, Bowen Li et al. presented the UAVDark135 [18] dataset and the ADTrack algorithm. Their work aimed to tackle the challenge of achieving reliable tracking of unmanned aerial vehicles (UAVs) under different lighting conditions. UAVDark135 is the inaugural benchmark specifically developed for tracking objects during nighttime. It consists of more than 125,000 frames that have been manually annotated, addressing a deficiency in current benchmarks. The paper details the ADTrack algorithm, a discriminative correlation filter-based tracker with illumination adaptive and anti-dark capabilities, utilizing image illuminance information and an image enhancer for real-time, all-day tracking. ADTrack performs better in both bright and dark environments—as evidenced by extensive testing on benchmarks such as UAV123@10fps [46], DTB70 [47], and UAVDark135—achieving over 30 FPS on a single CPU. While effective, the paper recommends broader comparisons with other state-of-the-art trackers and future research on image enhancement, multi-sensor fusion, and UAV hardware optimization.

3.14. DarkTrack2021

Junjie Ye et al. presented the DarkTrack2021 [19] dataset to tackle the difficulty of tracking unmanned aerial vehicles in low-light situations. The dataset consists of 110 annotated sequences containing more than 100,000 frames, providing a varied evaluation platform for tracking UAVs during nighttime. The researchers created an effective low-light enhancer called the Spatial-Channel Transformer (SCT), which combines a Spatial-Channel Transformer with a robust non-linear curve projection model to effectively enhance low-light images. The Spatial-Channel Attention Module (SCT) employs a technique that effectively combines global and local information, resulting in enhanced image quality by reducing noise and improving illumination in nighttime scenes. This study utilizes the proposed ADTrack algorithm together with 16 state-of-the-art handmade correlation filter (CF)-based trackers to evaluate their performance on tracking benchmarks UAV123@10fps, DTB70, and UAVDark135. The aim is to demonstrate the comprehensive robustness of the proposed ADTrack algorithm in all-day UAV tracking. Evaluations conducted on the public UAVDark135 and the new DarkTrack2021 benchmarks demonstrated that SCT exhibited superior performance compared to existing methods in tracking UAVs during nighttime. The practicality of the approach has been confirmed through real-world tests. The DarkTrack2021 dataset and SCT code are openly accessible on GitHub for additional research and experimentation.

3.15. BioDrone

Xin Zhao et al. presented the BioDrone [20] dataset. BioDrone is a pioneering visual benchmark for Single Object Tracking (SOT) that utilizes bionic drones. It specifically tackles the difficulties associated with tracking small targets that undergo significant changes in appearance, which are common in flapping-wing UAVs. The dataset consists of 600 videos containing 304,209 frames that have been manually labeled. Additionally, there are automatically generated labels for ten challenge attributes at the frame level. The study presents a new baseline method, UAV-KT, optimized from KeepTrack [48], and evaluates 20 SOT models, ranging from traditional approaches like KCF [49] to sophisticated models combining CNNs and SNNs. The results of comprehensive experiments demonstrate that UAV-KT outperforms other methods in handling challenging vision tasks with resilience. The paper emphasizes BioDrone's potential for advancing SOT algorithms and encourages future research to address remaining challenges, such as camera shake and dynamic visual environments.

In the Appendix A, a visual representation of each of the aforementioned UAV datasets is provided, from which further clarification can be obtained.

4. Data Diversity of UAV

The advent of UAVs has opened new frontiers in data collection and analysis, transforming numerous fields with their versatile applications. The datasets generated by UAVs are diverse, encompassing various data types and serving multiple purposes. This section provides an overview of the various uses of UAV datasets, examines their diversity, and explores the methods applied to utilize these datasets in different studies.

4.1. Overview of UAV Dataset Uses

UAV datasets are pivotal in numerous domains, including disaster management, surveillance, agriculture, environmental monitoring, and human behavior analysis. The unique aerial perspectives provided by UAVs enable the collection of high-resolution imagery and videos, which can be used for mapping, monitoring, and analyzing different environments and activities.

4.1.1. Disaster Management

UAV datasets are often used to figure out how much damage hurricanes, earthquakes, and floods have done. High-resolution images and videos captured by UAVs allow for precise mapping of affected areas and the identification of damaged infrastructure.

4.1.2. Surveillance

In urban and rural settings, UAV datasets support advanced surveillance activities. They facilitate the monitoring of traffic, detection of illegal activities, and overall urban planning by providing real-time, high-resolution aerial views.

4.1.3. Agriculture

UAV datasets help in monitoring crop health, assessing irrigation needs, and detecting pest infestations. Multispectral and hyperspectral imaging from UAVs enable detailed analysis of vegetation indices and soil properties.

4.1.4. Environmental Monitoring

UAVs are used to monitor forest health, wildlife, and water bodies. They provide data for studying ecological changes, tracking animal movements, and assessing the impacts of climate change.

4.1.5. Human Behavior Analysis

UAV datasets contribute to analyzing human activities and behaviors in public spaces. They are used for action recognition, pose estimation, and crowd monitoring, offering valuable insights for security and urban planning.

4.2. Variability of UAV Databases

The diversity of UAV datasets lies in their varied data types, capture conditions, and application contexts. This diversity ensures that UAVs can address a wide range of tasks, each requiring specific data characteristics.

4.2.1. Data Types

UAV datasets include RGB images, infrared images, depth maps, and multispectral and hyperspectral images [50]. To capture complex scenarios for human behavior analysis, the UAV-Human dataset, for example, combines RGB videos, depth maps, infrared sequences, and skeleton data.

4.2.2. Capture Conditions

A variety of conditions, such as different times of day, weather, light (low light or varied illumination), and flight altitudes, are encountered when gathering UAV datasets.

This variety makes sure that models trained on these datasets are strong and work well in a variety of settings.

4.2.3. Application Contexts

UAV datasets are tailored for specific applications. For example, visualizing data, object annotations, and flight data are used to address specific problems that come up when monitoring traffic from the air. Furthermore, the application of high-resolution images of damage taken after a disaster enables accurate assessment of the damage.

4.3. Methods Applied to the UAV Dataset

Various methods are applied to UAV datasets to extract valuable insights and solve specific problems. These methods include machine learning, computer vision techniques, and advanced data processing algorithms. In Tables 3 and 4, an overview of the methods used and the analysis of results are given to gain a better understanding.

Table 3. Summary of experimented methods and results on different datasets.

Dataset Name	Experimental Methods in Base Dataset Publication	Analysis of Results
RescueNet [6]	PSPNet, DeepLabv3+, Attention UNet, Segmenter [51]	Attention UNet achieved the best performance among all evaluated models. PSPNet showed better performance compared to DeepLabv3+ by using pyramid pooling. DeepLabv3+ provided moderate results, improving on the loss of boundary information. Segmenter showed varying results depending on the backbone (ViT-Tiny vs. ViT-Small), with heavier backbones achieving better results.
UAV-Human [7]	Guided Transformer I3D Network, Video Transformers, Full Model (Author's novel method)	Night-vision and IR videos outperformed previous findings in low-light conditions, achieving 28.72% and 26.56% accuracy, respectively. However, depth sequences face noise issues, and fisheye distortion impacts performance. In ablation studies, using KL Divergence Constraint resulted in 21.68% accuracy, while employing guidance loss and Video Transformers yielded 21.49% accuracy without RGB stream guidance. Overall, the full model had the highest accuracy among fisheye-based methods.
AIDER [8]	Novel networks (ERNet, SCFCNet, SCNet, baseNet), VGG16, ResNet50, MobileNet	The VGG16 model had the highest accuracy at 91.9% but a low frame rate of 2, while consuming 59.3 MB of memory. MobileNet had a high frame rate of 20 but lower accuracy at 88.5%. Custom networks like ERNet and SCFCNet had good accuracy at 90.1% and 87.7% with high frame rates of 53 and 76, making them suitable for real-time UAV applications.
AU-AIR [9]	YOLOv3-tiny, MobileNetv2-SSD Lite	YOLOv3-tiny achieved higher mAP (38.2%) and better FPS (22) compared to MobileNetv2-SSDLite (32.8% mAP and 19 FPS), highlighting its better performance for real-time object detection tasks using UAVs.
ERA [10]	VGG-16, DenseNet-121, NASNet-L, C3D [52]	DenseNet-121 achieved the highest overall accuracy (62.3%) among the models, followed by NASNet-L (60.2%) and VGG-16 (51.9%). The C3D models had the lowest accuracy (around 30%).

Table 4. Summary of experimented methods and results on different datasets.

Dataset Name	Experimented Methods on Dataset	Analysis of Results
UAVid [11]	FCN-8s [53], Dilation Net, U-Net [54], MS-Dilation Net	MS-Dilation Net achieved the highest mean IoU score of 57.3% with pre-training and feature space optimization, demonstrating the best performance among the models evaluated.
VRAI [11]	MGN, RAM, RNN-HA, Ensemble methods (e.g., ID Classification Loss, Triplet + ID Loss), Novel methods (Multi-task, Multi-task + Discriminative Parts)	The multi-task model with discriminative parts achieved the highest mAP (78.63%) and CMC-1 (80.30%). The models using Triplet + ID Loss also showed high performance, particularly with Resnet-101 and Resnet-152 backbones.
FOR-instance [13]	None, as the paper is solely focused on constructing the dataset and explaining how to utilize it for model.	N/A
VERI-Wild [14]	GoogLeNet [55], Triplet [56], Softmax [57], CCL [58], HDC [59], Unlabeled GAN [60,61], EN (Embedding Network with Triplet and Softmax Loss), FDA-Net ⊖ Att, FDA-Net	FDA-Net consistently outperforms the other models across different settings, achieving the highest mAP (35.11%) and match rate (R = 1 of 64.03% for small dataset). The proposed FDA-Net model demonstrates its effectiveness in vehicle re-identification tasks.
UAV-Assistant (UAVA) [15]	Singleshotpose [62], Direct, IoU-based experimental methods (e.g., I0.1, I0.2, I0.1-0.4, G0.1, S0.1, S0.2), Generalized IoU-based method (Gauss0.1) [63]	Gauss0.1 showed the best overall performance, particularly in the 6D Pose-5 and 6D Pose-10 metrics. Metrics such as NPE, OE, and CPE were used, with lower values indicating better performance and higher values for Acc5 and Acc10 indicating better performance.
KITE [16]	Baseline Systems (Unadapted System, Domain-Specific System), Domain Adaptation (Text-Only Adaptation, Rescoring), Multi-Modal Experiments (Text and Visual Information)	Domain adaptation and multi-modal approaches significantly improved the performance of speech recognition systems for UAV control. The Unadapted System had a WER of 56.2%, while the Domain-Specific System achieved 11.7%.
UAV-Gesture [17]	Pose-based CNN (P-CNN)	P-CNN achieved an overall accuracy of 91.9% for gesture recognition. The dataset included 119 video clips, 37,151 annotated frames, and 13 gestures, providing a robust resource for gesture and action recognition research.
DarkTrack2021 [19]	Novel ensemble method: SCT	The full implementation of SCT (Spatial-Channel Transformer) with all components enabled showed the highest improvement in tracking performance, with success rate and precision gains of 13.3% and 15.4%, respectively.
UAVDark135 [18]	ADTrack, State-of-the-art trackers (e.g., AutoTrack, SiamFC++, ARCF-HC, SiamRPN++)	ADTrack outperformed all other models in both bright and dark conditions, showing superior performance with the highest DP and AUC scores on the UAVDark135 dataset.
BioDrone [20]	KeepTrack, UAV-KT, Generic SOT Trackers	UAV-KT, designed for flapping-wing UAVs, showed a 5% improvement over KeepTrack in precision, normalized precision, and success scores. Generic SOT Trackers were compared for robustness and performance across various conditions.

4.3.1. Machine Learning and Deep Learning

Deep learning models, such as convolutional neural networks (CNNs) [64], are widely used for tasks like object detection, segmentation, and classification. Some examples are as follows:

- The RescueNet dataset employs models like PSPNet, DeepLabv3+, and Attention UNet for semantic segmentation to assess disaster damage.

- The UAVid Dataset presents deep learning baseline methods like Multi-Scale-Dilation net. The ERA dataset establishes a benchmark for event recognition in aerial videos by utilizing pre-existing deep learning models like the VGG models (VGG-16, VGG19) [29], Inception-v3 [65], the ResNet models (ResNet-50, ResNet-101, and ResNet-152) [30], MobileNet, the DenseNet models (DenseNet-121, DenseNet-169, and DenseNet-201) [35], and NASNet-L [66].

Traditional computer vision algorithms, alongside advanced machine learning and deep learning methods, have shown considerable effectiveness in UAV applications. For instance, ORB (Oriented FAST and Rotated BRIEF) and FLANN (Fast Library for Approximate Nearest Neighbors) have been employed for efficient object recognition and feature matching in UAV rescue systems, enabling real-time processing under hardware constraints [67]. Similarly, the SIFT (Scale-Invariant Feature Transform) algorithm has proven to be highly effective in image mosaicking tasks, ensuring high accuracy and computational efficiency, particularly in agricultural monitoring [68]. These algorithms provide efficient alternatives in resource-constrained environments, proving essential for tasks including object detection, feature matching, and image alignment in UAV applications. Algorithms like ORB, FLANN, and SIFT, now utilized in object detection and mosaicking, could be further refined and incorporated into future UAV datasets to tackle developing issues.

In the domain of deep learning, ensemble methods play a crucial role. They not only assess model performance but also boost accuracy while keeping the model's equilibrium intact, such as the following:

- In the VRAI dataset, ensemble techniques were utilized such as Triplet Loss, Contrastive Loss, ID Classification Loss, and Triplet + ID Loss, and multi-task and multi-task + discriminative parts were introduced. These ensemble methods performed better than the state-of-the-art methods in their claim.

4.3.2. Transfer Learning

Transfer learning is used to leverage pre-trained models on UAV datasets, allowing for quicker and more efficient training.

- Pre-trained YOLOv3-Tiny and MobileNetv2-SSDLite models, for example, are used for real-time object detection in the AU-AIR [9] dataset.

4.3.3. Event Recognition

UAVs have proven to be highly proficient in the field of event recognition and have gained significant popularity in this domain. Some examples are given below:

- The ERA dataset has been subjected to various methods for event recognition in aerial videos, including DenseNet-201 and Inception-v3. These methods have demonstrated notable accuracy in identifying dynamic events from UAV footage.
- The BioDrone dataset assesses single object tracking (SOT) models and investigates new optimization approaches for the cutting-edge KeepTrack method for robust vision, which is presented by flapping-wing unmanned aerial vehicles [20].

4.3.4. Multimodal Analysis

Combining data from multiple sensors enhances the analysis capabilities of UAV datasets. The multimodal approach of the UAV-Human dataset, which combines RGB, infrared, and depth data, makes a thorough analysis of human behavior possible.

4.3.5. Creative Algorithms

New algorithms are created to tackle particular problems in the analysis of data from unmanned aerial vehicles. Some examples are as follows:

- The UAV-Gesture [17] dataset employs advanced gesture recognition algorithms to enable UAV navigation and control based on human gestures.

- The UAVDark135 [18] makes use of ADTrack, a tracker that adapts to varying lighting conditions and makes use of discriminative correlation filters. It also has anti-dark capabilities.
- To address the issue of fisheye video distortions, the authors of the UAV-Human [7] dataset suggest a fisheye-based action recognition method that uses flat RGB videos as guidance.
- To classify disaster events from a UAV, the authors of the AIDER [8] dataset created a lightweight (CNN) architecture that they named ERNet.
- VERI-Wild [14] introduces FDA-Net, a novel method for vehicle identification. It includes an embedding discriminator and a feature distance adversary network to enhance the model's capacity to differentiate between various automobiles.

4.3.6. Managing Diverse Conditions

Various environmental conditions, such as different lighting, weather, and occlusions, present challenges that are often addressed by methodologies. For example, DarkTrack2021 used the low-light enhancer-based method SCT to handle performance in low-light conditions.

The diversity of UAV datasets is a cornerstone of their utility, enabling a wide array of applications across different fields. From disaster management to human behavior analysis, the rich variety of data types, capture conditions, and application contexts ensures that UAV datasets can meet the specific needs of each task. The application of advanced methods, including deep learning, transfer learning, and multimodal analysis, further enhances the value derived from these datasets, pushing the boundaries of what UAVs can achieve in research and practical applications.

5. The Potential of Computer Vision Research in UAV Datasets

UAVs have greatly expanded the fields of computer vision research. UAV datasets offer unique and flexible data that are used in a range of computer vision tasks, from recognizing actions to finding objects. This section explores how UAV datasets are advancing computer vision research and contributing to various tasks from action recognition to object detection, as illustrated in Figure 2, which highlights the diverse applications and the development of new methods centered around these datasets. Table 5 presents the use cases of methods on UAV datasets identified during our survey of the UAV datasets.

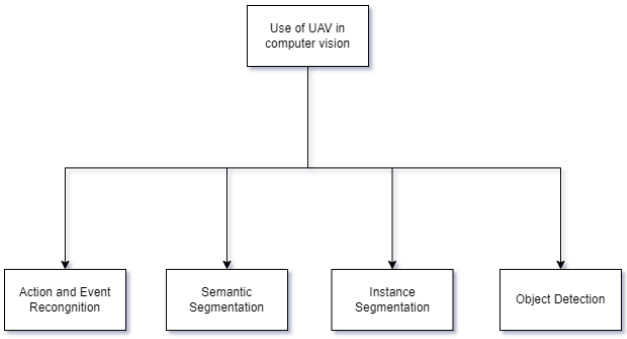


Figure 2. Diverse applications of UAV datasets in computer vision research.

Table 5. Summary of methods employed on UAV datasets and their benefits.

Employed Method	Name of the Dataset	Benefit from the Use of Method
Attention UNet28, ViT-Tiny, ViT-Small	RescueNet [6]	Improved disaster response strategies and enhanced model performance in segmentation tasks through transfer learning
Fisheye-based action recognition approach, HigherHRNet, AlphaPose	UAV-Human [7]	Robust models for human behavior understanding
ERNet	AIDER [8]	High performance with minimal memory requirements, suitable for real-time aerial image classification
YOLOv3-Tiny, MobileNetv2-SSDLite	AU-AIR [9]	Real-time object detection on UAVs, bridging the gap between computer vision and robotics
DenseNet-201, I3D-Inception-v1, TRN-Inception-v3	ERA [10]	High performance in single-frame and video classification tasks
Multi-Scale-Dilation net, FSO, 3D CRF	UAVid [11]	Enhanced semantic segmentation performance in urban scenes, addressing large-scale variation and moving object recognition
Convolutional and connection layers, weight matrices, weighted pooling	VRAI [12]	Superior vehicle re-identification performance
Aggregating tree-wise F1 scores, weighting coefficients for averaging F1 scores	FOR-instance [13]	Improved methods for individual tree segmentation, crucial for understanding forest ecosystems
FDA-Net (Feature Distance Adversary Network)	VERI-Wild [14]	Enhanced discriminative capability in vehicle re-identification tasks
Smooth silhouette loss	UAV-Assistant (UAVA) [15]	Improved performance in 3D pose estimation tasks
Time-delay neural network, domain adaptation techniques	KITE [16]	Enhanced UAV command recognition systems through visual context and domain adaptation
Pose-based Convolutional Neural Network (P-CNN)	UAV-Gesture [17]	High accuracy in gesture recognition for UAV control
Spatial-Channel Transformer, curve projection model	DarkTrack2021 [19]	Improved nighttime UAV tracking accuracy by enhancing low-light images
Illumination adaptive, anti-dark capabilities, efficient image enhancer	UAVDark135 [18]	Superior performance in all-day aerial object tracking, adaptability to different light conditions
KeepTrack-optimized UAV-KT	BioDrone [20]	Addresses challenges in tracking tiny targets with drastic appearance changes, providing a robust benchmark for vision research

5.1. Leveraging UAV Datasets for Computer Vision Applications

Human behavior analysis, emergency response, tracking at night, surveillance, and many other uses can be achieved with UAV datasets in computer vision. These are some of the areas where UAV datasets are used, along with an example of how to describe a dataset based on the datasets we talked about in our research paper.

5.1.1. Human Behavior Understanding and Gesture Recognition

The UAV-Human platform is essential for utilizing UAVs to study human behavior, including a range of conditions and perspectives for pose estimation and action recognition. This dataset contains multi-modal information, including skeleton, RGB, infrared, and night vision modalities. Essential for UAV control and gesture identification, UAV-Gesture contains 119 high-definition video clips with 13 gestures for command and navigation that are marked with body joints and gesture classes. Because this dataset was captured outside, it has more practical UAV control applications because of the variations in phase, orientation, and body shape.

5.1.2. Emergency Response and Disaster Management

RescueNet provides detailed pixel-level annotations and high-resolution images for 10 classes, including buildings, roads, pools, and trees. It is designed for post-disaster damage assessment using UAV imagery. It supports semantic segmentation using state-of-the-art models, enhancing natural disaster response and recovery strategies. AIDER focuses on classifying disaster events, utilizing images of traffic accidents, building collapses, fires, and floods to support real-time disaster management applications by training convolutional neural networks.

5.1.3. Traffic Surveillance and Vehicle Re-Identification

In traffic surveillance, AU-AIR prioritizes real-time performance and offers annotations for a variety of object categories, including cars, buses, and pedestrians. It bridges the gap between computer vision and robotics by offering multi-modal sensor data for advanced research in data fusion applications. VRAI is the largest UAV-based vehicle re-identification dataset, containing over 137,613 images of 13,022 vehicles with annotations for unique IDs, color, vehicle type, attributes, and discriminative parts. It supports vehicle ReID tasks with diverse scenarios and advanced algorithms. VERI-Wild, which contains over 400,000 photos of 40,000 vehicles taken by 174 CCTV cameras in various urban settings, is essential for research on vehicle re-identification. It uses techniques like FDA-Net to improve ReID accuracy by addressing variations in backgrounds, illumination, occlusion, and viewpoints.

5.1.4. Event Recognition and Video Understanding

For training models in event recognition in UAV videos, ERA contains 2864 labeled video snippets for twenty-four event classes and one normal class that were gathered from YouTube. This dataset captures dynamic events in various conditions, supporting temporal event localization and video retrieval tasks.

5.1.5. Nighttime Tracking and Low-Light Conditions

Including 110 annotated sequences with over 100,000 frames, DarkTrack2021 is crucial for improving UAV tracking at night. By employing Spatial-Channel Transformers (SCT) and non-linear curve projection models, it improves the quality of low-light images and offers a thorough assessment framework. The UAVDark135 dataset and the ADTrack algorithm are designed for all-day aerial tracking. ADTrack performs well in low light and adjusts to various lighting conditions thanks to its discriminative correlation filter foundation. More than 125,000 frames, specially annotated for low-light tracking scenarios, are included in the UAVDark135 dataset.

5.1.6. Object Tracking and Robust Vision

With 600 videos and 304,209 manually labeled frames, BioDrone is a benchmark for single object tracking with bionic drones. It captures challenges such as camera shake and drastic appearance changes, supporting robust vision analyses and evaluations of various single object tracking algorithms.

5.1.7. Urban Scene Segmentation and Forestry Analysis

UAVid provides annotations for eight classes and 30 high-resolution video sequences in 4K resolution to address segmentation challenges in urban scenes. It uses models such as Multi-Scale-Dilation net to support tasks like population density analysis and traffic monitoring. FOR-instance provides UAV-based laser scanning data for tree instance segmentation and is intended for use in point cloud segmentation in forestry. It facilitates benchmarking and method development by supporting both instance and semantic segmentation.

5.1.8. Multimodal Data Synthesis and UAV Control

UAV-Assistant facilitates monocular pose estimation by introducing a multimodal dataset featuring exocentric and egocentric views. It enhances 3D pose estimation tasks with a novel smooth silhouette loss function and differentiable rendering techniques. KITE incorporates spoken commands, audio, and images to enhance UAV control systems. It includes commands recorded by 16 speakers, supporting movement, camera-related, and scenario-specific commands with multi-modal approaches.

Together, these datasets improve a wide range of computer vision applications, including robust vision in difficult conditions, real-time traffic surveillance, emergency response, and human behavior analysis.

5.2. Development of Novel Methods Using UAV Datasets

UAV datasets have spurred the development of innovative methods in computer vision. As an example, the Guided Transformer I3D framework, which addresses distortions through unbounded transformations guided by flat RGB videos, was developed using the UAV-Human dataset. This framework enhances action recognition performance in fisheye videos. This approach is a prime example of how UAV datasets drive the creation of specialized algorithms to address particular difficulties brought about by aerial viewpoints.

The DarkTrack2021 benchmark introduces a Spatial-Channel Transformer (SCT) for enhancing low-light images in nighttime UAV tracking. Meanwhile, Bowen Li et al. present the UAVDark135 dataset and the ADTrack algorithm for all-day aerial object tracking. ADTrack, equipped with adaptive illumination and anti-dark capabilities, outperforms other trackers in both well-lit and dark conditions. It processes over 30 frames per second on a single CPU, ensuring efficient tracking under various lighting conditions. The study emphasizes how crucial image illuminance data are and suggests a useful image enhancer to improve tracking performance in all-day situations.

For emergency response applications, the AIDER dataset has facilitated the development of ERNet, a lightweight CNN architecture optimized for embedded platforms. ERNet's architecture, which incorporates downsampling at an early stage and efficient convolutional layers, allows for real-time classification of aerial images on low-power devices. This showcases the practical use of UAV datasets in disaster management.

The VERI-Wild dataset introduces a novel approach called FDA-Net for vehicle reidentification. This method utilizes a unique type of network to generate difficult negative examples in the feature space. On the other hand, the VRAI dataset developed a specialized vehicle ReID algorithm that leverages detailed annotation information to explicitly identify unique parts for each vehicle instance in object detection.

Ultimately, UAV datasets are essential in the field of computer vision research, providing distinct data that are invaluable for a diverse array of applications. They allow for the development of novel methods tailored to the specific challenges and opportunities presented by UAV technology, accelerating progress in areas such as human behavior analysis, emergency response, and nighttime tracking.

6. Limitations of UAVs

While Unmanned Aerial Vehicles have significantly advanced data collection and analysis in numerous fields, they are not without limitations, particularly concerning the datasets they generate. This section delves into the primary constraints associated with UAV datasets, emphasizing their impact on the field and suggesting areas for improvement.

6.1. Data Quality and Consistency

One of the most pressing limitations of UAV datasets is the inconsistency in data quality. Weather, time of day, and UAV stability are just a few variables that can affect the quality of data that UAVs collect. For example, datasets collected during poor weather conditions or at night may need more visibility and increased noise, complicating subsequent analysis and model training. Even with advancements like low-light image enhancers and

specialized algorithms for nighttime tracking, these solutions often need improvement and require further refinement to match the reliability of daytime data.

6.2. Limited Scope and Diversity

UAV datasets often need more diversity in terms of geographic locations, environmental conditions, and the variety of captured objects. Many existing datasets, such as AU-AIR and ERA, focus heavily on specific scenarios like urban traffic surveillance or disaster response, which limits their generalizability to other contexts. Additionally, datasets such as UAV-Human and UAVDark135 tend to feature limited subject diversity and controlled environments, which may not accurately represent real-world conditions. This lack of diversity can lead to models that perform well in specific conditions but struggle in untested environments.

6.3. Annotation Challenges

The process of annotating UAV datasets is often time-consuming and labor-intensive. High-resolution images and videos captured by UAVs require detailed, pixel-level annotations, which are essential for tasks like semantic segmentation and object detection. This is clearly seen in datasets such as RescueNet and FOR-Instance, where the annotation process is recognized as a major bottleneck. The intensive labor required for comprehensive annotation limits the availability of large, well-labeled datasets, which are crucial for training robust machine learning models.

6.4. Computational and Storage Demands

The high resolution and large volume of data generated by UAVs pose significant computational and storage challenges. Processing and analyzing large-scale UAV datasets demand substantial computational resources and advanced hardware, which may only be readily available to some researchers. For example, the dense and high-resolution images in datasets like UAVid and BioDrone require extensive processing power for effective utilization. Additionally, the storage of such vast amounts of data can be impractical for some institutions, hindering widespread access and collaboration.

6.5. Integration with Other Data Sources

Another limitation is the integration of UAV datasets with other data sources. While multimodal datasets that combine UAV data with other sensor inputs (such as satellite imagery, GPS data, and environmental sensors) provide richer insights, they also introduce complexity in data alignment and fusion. The AU-AIR dataset, which includes visual data along with GPS coordinates and IMU data, exemplifies the potential and challenges of such integration. Ensuring the synchronized and accurate fusion of data from multiple sources remains a technical hurdle that needs addressing.

6.6. Real-Time Data Processing

The ability to process and analyze UAV data in real-time is critical for applications like disaster response and surveillance. However, achieving real-time processing with high accuracy is challenging due to the aforementioned computational demands. Models such as those evaluated in the DarkTrack2021 and UAVDark135 datasets show promise but often require optimization to balance speed and accuracy effectively. Real-time processing also necessitates robust algorithms capable of handling dynamic environments and changing conditions without significant delays.

6.7. Ethical and Legal Considerations

Finally, the use of UAVs and their datasets is subject to various ethical and legal considerations. Issues such as privacy, data security, and regulatory compliance must be addressed to ensure responsible and lawful use of UAV technology. These considerations

can limit the scope of data collection and usage, particularly in populated areas or sensitive environments, thereby constraining the availability and applicability of UAV datasets.

Despite the transformative potential of UAV datasets across various disciplines, their limitations must be acknowledged and addressed to maximize their utility. Improving data quality, enhancing dataset diversity, streamlining annotation processes, and overcoming computational and storage challenges are essential steps. Additionally, integrating UAV data with other sources, advancing real-time processing capabilities, and adhering to ethical and legal standards will ensure that UAV datasets can be effectively leveraged for future research and applications. By tackling these limitations, the field can fully harness the power of UAV technology to drive innovation and deepen our understanding of complex, dynamic environments from an aerial perspective.

7. Prospects for Future UAV Research

Future studies on UAV datasets need to focus on a few crucial areas to improve their usefulness and cross-domain applicability as the field grows. The following suggestions highlight the crucial paths for creating UAV datasets and maximizing their potential for future innovations.

7.1. Enhancing Dataset Diversity and Representativeness

Further investigations ought to concentrate on generating more varied and representative UAV datasets. This involves capturing data in a wider range of environments, weather conditions, and geographic locations to ensure models trained on these datasets are robust and generalizable. To obtain comprehensive data for tasks like environmental monitoring, urban planning, and disaster response, datasets can be expanded to include a variety of urban, rural, and natural settings.

7.2. Incorporating Multimodal Data Integration

Integrating multiple data modalities, such as thermal, infrared, LiDAR [66], and hyperspectral [50] imagery, can significantly enrich UAV datasets. In the future, these data types should be combined to create multimodal datasets that provide a more comprehensive view of the scenes that were recorded. This integration can improve the accuracy of applications such as vegetation analysis, search and rescue operations, and wildlife monitoring.

7.3. Advancing Real-Time Data Processing and Transmission

For applications like emergency response and traffic monitoring that demand quick analysis and decision-making, developing techniques for real-time data processing and transmission is essential. Future research should focus on optimizing data compression, transmission protocols, and edge computing techniques to enable swift and efficient data handling directly on UAVs.

7.4. Improving Annotation Quality and Efficiency

High-quality annotations are vital for the effectiveness of UAV datasets in training machine learning models. Future studies should investigate automated and semi-automated annotation tools that leverage AI to reduce manual labor and improve annotation accuracy. Additionally, crowdsourcing and collaborative platforms can be utilized to gather diverse annotations, further enhancing dataset quality.

7.5. Addressing Ethical and Privacy Concerns

As UAVs become more prevalent, addressing ethical and privacy issues becomes increasingly important. Guidelines and frameworks for the ethical use of UAV data should be established by future research, especially for applications involving surveillance and monitoring. It is important to focus on creating methods that protect privacy and collect data in a way that respects regulations and earns the trust of the public.

7.6. Expanding Application-Specific Datasets

The creation of customized datasets for specific uses can effectively boost new ideas in certain areas. For instance, datasets focused on agricultural monitoring, wildlife tracking, or infrastructure inspection can provide domain-specific insights and improve the precision of related models. To address the specific needs of various industries, future research should give priority to developing such targeted datasets.

7.7. Enhancing Interoperability and Standardization

Standardizing data formats and annotation protocols across UAV datasets can make it easier for researchers and developers to use and make the datasets more interoperable. Future efforts should aim to establish common standards and benchmarks, enabling the seamless integration of datasets from various sources and promoting collaborative research efforts.

7.8. Utilizing Advanced Machine Learning Techniques

The application of cutting-edge machine learning techniques, such as deep learning and reinforcement learning, to UAV datasets holds immense potential for advancing UAV capabilities. Future research should explore innovative algorithms and models that can leverage the rich data provided by UAVs to achieve breakthroughs in areas like autonomous navigation, object detection, and environmental monitoring.

7.9. Leveraging Advanced Machine Learning Techniques

Longitudinal studies that collect UAV data over long periods of time can provide useful information about how things change over time in different settings. Future research should emphasize continuous data collection efforts to monitor changes in ecosystems, urban developments, and disaster-prone areas, enabling more informed and proactive decision making.

7.10. Fostering Collaborative Research and Open Data Initiatives

Encouraging collaboration among researchers, institutions, and industries can accelerate advancements in UAV datasets. Open data initiatives that make UAV datasets public should be supported by future research. These initiatives will encourage innovation and allow a wider range of researchers to contribute to and use these resources.

By addressing these future research directions, the field of UAV datasets can continue to evolve, offering increasingly sophisticated tools and insights that drive progress across multiple domains. UAV datasets are still being improved and added to, which is very important for getting the most out of UAV technology and making room for new discoveries and uses.

8. Results and Discussion of Reviewed Papers

The datasets discussed in this section represent the application of the papers reviewed in this survey. Our analysis of the datasets revealed that KITE, RescueNet, and Biodrone are relatively new and have not been thoroughly investigated in the literature. While one of the datasets we reviewed, ERA, is not very recent, it still lacks a sufficient amount of study to fully emphasize its potential. The datasets included in our review were selected based on the number of citations their associated papers have received, emphasizing those with higher citation counts. We delved into several papers that make compelling use of the datasets we evaluated. In our examination, we carefully reviewed the details of the analysis of results and experiments conducted by other researchers. These researchers utilized the datasets we assessed as benchmarks and applied various methods. We have included the best results for the methods applied to the datasets we reviewed in this section and in Tables 6–8.

Table 6. Performance metrics and results for different datasets and methods.

Dataset Name	Reference	Methods	Performance
AU-AIR [9]	[69]	YOLOv3	mAP 59.83 Speed (FPS) 29
		YOLOv4	mAP 67.35 Speed (FPS) 24
		RSSD-TA-LSTM-GID	mAP 71.68 Speed (FPS) 23
	[70]	res2net50	mAP 88.93 Speed (FPS) 45.73
		rs2net101	mAP 90.52 Speed (FPS) 7.21
		hourglass-104	mAP 91.62 Speed (FPS) 7.19
	[71]	RetinaNet	Voting Strategy Unanimous mAP (%) 6.63
		YOLO + RetinaNet	Voting Strategy Consensus mAP (%) 3.69
		RetinaNet + SSD	Voting Strategy Consensus mAP (%) 4.03
	[72]	Faster R-CNN	mAP (%) 13.77
		SSD	mAP (%) 9.1
		YOLOv3	mAP (%) 13.33
		YOLOv4	mAP (%) 25.94
FOR-instance [13]	[73]	PointNet	mIoU 35.65 micro F1 52.56
		PointNet++	mIoU 33.00 micro F1 49.57
		Point Transformers	mIoU 22.97 micro F1 37.13
	[74]	HFC (on CULS plot [1])	Precision 0.89 Recall 0.8 F1 score 0.84
		HFC (on NIBIO plot)	Precision 0.89 Recall 0.85 F1 score 0.87
		HFC (on NIBIO2 plot)	Precision 0.85 Recall 0.85 F1 score 0.85
		HFC (on SCION plot)	Precision 0.95 Recall 0.90 F1 score 0.92
		HFC (on RMIT plot)	Precision 0.89 Recall 0.85 F1 score 0.87
		HFC (on TUWEIN plot)	Precision 0.84 Recall 0.80 F1 score 0.82

Table 6. *Cont.*

Dataset Name	Reference	Methods	Performance
UAV-Assistant [15]	[75]	BPnP [76]	ACC2 ACC5 95.2 98.36 55.31 85.34
		HigherHRNet [25]	ACC2 ACC5 89.92 97.75
		HRNet [77]	ACC2 ACC5 90.75 98.04

[1]: "Plot" here refers to the forest types that were looked at in the FOR-instance dataset release.

Table 7. Performance metrics and results for different datasets and methods.

Dataset Name	Reference	Methods	Performance
AIDER [8]	[78]	EmergencyNet	memory (MB) F1 Score (%) 0.368 95.7
		VGG16	memory (mB) F1 Score (%) 59.39 96.4
		ResNet50	memory (MB) F1 Score (%) 96.4 96.1
	[79]	AISCC-DE2MS	MSE PSNR 0.042 61.898
		Genetic Algorithm	MSE PSNR 0.06 60.349
		Cat Swarm Algorithm	MSE PSNR 0.12 57.339
		Artificial Bee Colony Algorithm	MSE PSNR 0.165 55.956
DarkTrack2021 [19]	[80]	SAM-DA-Track	AUC Precision (normalized) Precision 0.451 0.524 0.593
		UDAT	AUC Precision (normalized) Precision 0.421 0.499 0.570
		SiamBAN	AUC Precision (normalized) Precision 0.422 0.491 0.566
	[81]	SiamAPN	DP NDP AUC 0.43 0.389 0.446
		SiamAPN++	DP NDP AUC 0.494 0.446 0.375
UAV-Human [7]	[82]	Proposed Novel Method	Precision Recall F1 Score 0.49 0.49 0.48
	[83]	CLIP [84]	Top1/Top5 (Filtering ratio 90%) 1.79/7.05
		ViFi CLIP [85]	Top1/Top5 (Filtering ratio 90%) 4.67/15.18
	[86]	2s-MS&TA-HGCN-FC (Novel method)	CSv1 CSv2 44.33 70.69
		4s-MS&TA-HGCN-FC (Novel method)	CSv1 CSv2 45.72 71.84
		FR-AGCN [87]	CSv1 CSv2 43.98 69.5

Table 7. *Cont.*

Dataset Name	Reference	Methods	Performance		
UAVDark135 [18]	[88]	DCPT	Success Rate 0.577	Precision 0.703	Normalized Precision 0.701
		DIMP50-SCT	Success Rate 0.562	Precision 0.717	Normalized Precision 0.71
		DIMP18 [89]	Success Rate 0.542	Precision 0.702	Normalized Precision 0.69
	[90]	DL+SiamAPN	Success Rate 0.389	Precision 0.516	
		SiamAPN [91]	Success Rate 0.3	Precision 0.424	
		DL+DIMP50	Success Rate 0.544	Precision 0.7	
		DIMP50 [19]	Success Rate 0.526	Precision 0.672	
VRAI [12]	[92]	Proposed Novel Method	mAP 0.828	R-1 Accuracy 0.844	
		TransReID [93]	mAP 0.814	R-1 Accuracy 0.826	
	[93]	RotTrans	mAP 0.848	R-1 Accuracy 0.838	
		TransReID	mAP 0.786	R-1 Accuracy 0.803	

DP: Dynamic Precision; NDP: Normalized Dynamic Precision; Top1/Top5: The authors of [83] employed the metric of Top1/Top5 for measuring accuracy in single-label classification.

Table 8. Performance metrics and results for different datasets and methods.

Dataset Name	Reference	Methods	Performance
UAV-Gesture [17]	[94]	Novel Multifeature+CNN method	Accuracy 0.95
		P-CNN [45]	Accuracy 0.91
		MLP_7j [95]	Accuracy 0.94
	[95]	DD-Net_7j [96]	Accuracy 0.915
		P-CNN	Accuracy 0.919
		MLP_7j	Accuracy 0.948
UAVid [11]	[97]	BANet	mIoU (%) 64.6
		MSD benchmark [11]	mIoU (%) 57.0
	[98]	A^2-FPN	mIoU (%) 65.7
		MSD benchmark	mIoU (%) 57.0

Table 8. *Cont.*

Dataset Name	Reference	Methods	Performance
	[99]	UNetFormer	mIoU (%) 67.8
		ABCNet	mIoU (%) 63.8
		BANet	mIoU (%) 64.6
		BoTNet	mIoU (%) 63.2
	[100]	MSD benchmark	mIoU (%) 57.0 FPS 1.00
		BiSeNet [101]	mIoU (%) 61.5 FPS 11.08
		CAN	mIoU (%) 63.5 FPS 15.14
VERI-Wild [14]	[102]	FDA-Net [14]	mAP (small) 0.351 mAP (medium) 0.298 mAP (large) 0.228
		PVEN	mAP (small) 0.825 mAP (medium) 0.77 mAP (large) 0.697
	[103]	MLSL [104]	mAP (large) 0.366 R-1 accuracy (large) 0.775
		FastReID	mAP (large) 0.773 R-1 accuracy (large) 0.925
	[105]	GiT	mAP (T10000) 0.675 R-1 accuracy (T10000) 0.854
		PCRNet [106]	mAP (T10000) 0.671 R-1 accuracy (T10000) 0.85
	[107]	HPGN	mAP (T10000) 0.65 R-1 accuracy (T10000) 0.8268
		Triplet Embedding [108]	mAP (T10000) 0.516 R-1 accuracy (T10000) 0.699
	[109]	Baseline [110]	mAP (large) 0.65 R-1 accuracy (large) 0.95
		SAVER	mAP (large) 0.677 R-1 accuracy (large) 0.958

8.1. AU-AIR

In their study, Jiahui et al. [69] selected AU-AIR as a benchmark dataset to create their proposed real-time object detection model, RSSD-TA-LATM-GID, specifically designed for small-scale object detection. The performance of their model surpassed that of YOLOv4 [111] and YOLOv3 [33]. The researchers employed the MobileNetv-SSDLite ensemble approach, which yielded the lowest mean average precision (mAP) score.

Walambe et al. [71] employed baseline models on the AU-AIR dataset as one of their evaluative benchmarks. The objective of the study was to demonstrate the attainability of different techniques and ensemble techniques in the detection of objects with varying scales. The baseline technique yielded the highest performance, with a mean average precision (mAP) score of 6.63%. This outcome was achieved by employing color-augmentation on the dataset. The performance metrics for the ensemble methods YOLO+RetinaNet and RetinaNet+SSD were found to be 3.69% and 4.03%, respectively. The authors, Saeed et al. [70], made modifications to the architecture of the CenterNet model by using other Convolu-

tional Neural Networks (CNNs) as backbones, such as resnet18, hourglass-104, resnet101, and res2net101. The findings are presented in Table 6.

In their paper [72], Gupta and Verma utilized the AU-AIR data as a reference point, employing a range of advanced models to achieve precise and automated detection and classification of road traffic. The YOLOv4 model achieved the highest mean average precision (mAP) score of 25.94% on the AU-AIR dataset. The Faster R-CNN and YOLOv3 models achieved the second and third highest maximum average precision (mAP) scores, with values of 13.77% and 13.33%, respectively.

8.2. FOR-Instance

Bountos et al. extensively utilized the "FOR-Instance" dataset in their study [73], while introducing their innovative approach FoMo-Net. The dataset was utilized to analyze point cloud representations obtained from LiDAR sensors in order to gain a deeper understanding of tree geometry. Existing baseline techniques such as PointNet, PointNet++, and Point Transformer were employed to accomplish these objectives on aerial modality. The corresponding findings are presented in Table 7. In a separate paper, Zhang et al. [74] used the "FOR-instance" dataset to train their proposed HFC algorithm and compare its performance with other established approaches. The authors utilized several techniques and ensemble approaches (Xing2023, HFC+Xing2023, HFC+Mean Shift, HFC) on several forest types (CULS, NIBIO, NIBIO2, SCION, RMIT, TUWIEN) shown in the FOR-instance dataset. Among all the methods, HFC demonstrated superior performance. The optimal outcomes achieved by the HFC approach on various forest types represented in the FOR-instance dataset are presented in Table 7.

8.3. UAV-Assistant

Albanis et al. used the UAV-Assistant dataset as the benchmark for their research [75]. They conducted a comparative analysis of BPnP [76] and HigherHRNet's [25] 6DOF object pose estimation using several different criteria. Analysis revealed that loss functions play a crucial role in posture estimation. Specifically, the l_p loss function outperformed the l_h loss function, particularly in the case of the M2ED drone, resulting in improved accuracy metrics. HigherHRNet demonstrated greater performance compared to HRNet [77] on smaller objects such as the Tello drone, but not on the M2ED drone, indicating its potential superiority under smaller object classifications. Their analysis of qualitative heatmaps revealed that the l_p loss function performed better than the Gaussian-distributed l_h model in accurately locating keypoints. Table 7 displays the accuracy metrics (ACC2 and ACC5) obtained from the research conducted by Albanis and his colleagues. In the case of BPnP, we have included the accuracy for both M2ED and Tello drones, respectively, as they achieved the highest accuracy outcomes. Regarding HRNet and HigherHRNet, they achieved the best accuracy specifically for M2ED.

8.4. AIDER

The AIDER dataset was utilized as a benchmark by Alrayes et al. and the authors of AIDER in developing their innovative method, "EmergencyNet". In their paper [78], various pre-trained models were applied to the AIDER dataset, with the best F1 accuracy achieved using VGG16 (96.4%) and ResNet50 (96.1%). However, the memory consumption for VGG16 and ResNet50 was quite high, at 59.39 MB and 96.4 MB, respectively. However, EmergencyNet achieved 95.7% F1 accuracy with only 0.368 MB of RAM. ResNet50 had nearly 24 million parameters while VGG16 had 14.8 million. Alrayes et al. benchmarked their AISCC-DE2MS model with AIDER. They found that their algorithm outperformed the genetic, cat-swarm, and artificial bee colony algorithms. MSE and PSNR were utilized to evaluate the model. These methods were used to compare five photos to evaluate the model's performance. The best result from the five photos is shown in Table 6.

8.5. DarkTrack2021

Changhong Fu and his team utilized the DarkTrack2021 benchmark as a foundation for developing the Segment Anything Model (SMA)-powered framework, SAM-DA. Their research [80] focused on effectively addressing illumination variation and low ambient intensity. They conducted a comparative analysis between their model and various methods, particularly the Baseline tracker UDAT [112] method. Their novel approach outperformed the Baseline UDAT method, achieving substantial improvements of 7.1% in illumination variation and 7.8% in low ambient intensity. The authors evaluated 15 state-of-the-art trackers and found that SAM-DA demonstrated the most promising results. Additionally, Changhong Fu delved into Siamese Object Tracking in their another study [81], highlighting the significance of UAVs in visual object tracking. They also leveraged the DarkTrack2021 datasets as a benchmark to assess model performance in low-illumination conditions, with detailed results and the applied models presented in Table 6.

8.6. UAV-Human

Azmat et al. [82] addressed UAV-captured data-based human action recognition (HAR) challenges and approaches in their UAV-Human dataset research. Azmat et al. evaluated their HAR system on 67,428 video sequences of 119 people in various contexts from the UAV-Human dataset. The approach has a mean accuracy of 48.60% across eight action classes, indicating that backdrops, occlusions, and camera motion hinder human movement recognition in this dataset. Lin et al. [83] studied text bag filtering techniques for model training, emphasizing data quality. Their ablation study indicated that text bag filtering ratio influences CLIP matching accuracy and zero-shot transfer performance. Filtering training data improved model generalization, especially in unsupervised learning. Huang et al. [86] evaluated the 4s-MS&TA-HGCN-FC skeleton-based action recognition model on the UAV-Human dataset. The model achieved 45.72% accuracy on the CSv1 benchmark and 71.84% on the CSv2 test, surpassing previous state-of-the-art techniques. They found that their technique can manage viewpoints, motion blurring, and resolution changes in UAV-captured data.

8.7. UAVDark135

Zhu et al. [88] and Ye et al. [90] used the UAVDark135 dataset to evaluate their strategies for increasing low-light tracking performance. The Darkness Clue-Prompted Tracking (DCPT) approach by Zhu et al. showed considerable gains, reaching a 57.51% success rate on UAVDark135. A 1.95% improvement over the base tracker demonstrates the effectiveness of including darkness clues. Additionally, DCPT's gated feature aggregation approach increased success score by 2.67%, making it a reliable nighttime UAV tracking system. Ye et al.'s DarkLighter (DL) approach improved tracking performance on the UAVDark135 dataset. DL improved SimpAPN [91,113] tracker's AUC by over 29% and precision by 21%. It also worked well across tracking backbones, enhancing precision and success rates in light variation, quick motion, and low-resolution circumstances. DL surpassed modern low-light enhancers like LIME by 1.68% in success rate and 1.45% in precision.

8.8. VRAI

VRAI was utilized to establish a vehicle re-identification baseline. Syeda Nyma Ferdous, Xin Li, and Siwei Lyu [92] tested their uncertainty-aware multitask learning framework on this dataset and achieved 84.47% Rank-1 accuracy and 82.86% mAP. This model's capacity to handle aerial image size and position fluctuations was greatly improved by multiscale feature representation and a Pyramid Vision Transformer (PVT) architecture. Shuoyi Chen, Mang Ye, and Bo Du [93] focused on vehicle ReID using VRAI. RotTrans, a rotation-invariant vision transformer, surpassed current innovative approaches by 3.5% in Rank-1 accuracy and 6.2% in mean average precision (mAP). This approach solved UAV-based vehicle ReID challenges that typical pedestrian ReID methods struggle with.

The process was further complicated by the need to present results in a certain format for performance evaluation.

8.9. UAV-Gesture

Usman Azmat et al. [94] and Papaioannidis et al. [95] utilized the UAV-Gesture dataset to evaluate their recommendations for human action and gesture recognition. They used the UAV-Gesture collection of 119 high-definition RGB movies representing 13 unique motions used to control UAVs. The dataset is ideal for testing recognition systems due to its diversity of views and movement similarities. The Usman Azmat et al. method achieved 0.95 action recognition accuracy on the UAV-Gesture dataset. Mean precision, recall, and F1-score for the system were 0.96, 0.95, and 0.94. Several investigations supported by confusion matrices showed the system's ability to distinguish gestures. Papaioannidis et al. found that their gesture recognition method outperformed DD-Net [96] and P-CNN [45] by 3.5% in accuracy. The authors stressed the need of using 2D skeletal data from movies to boost recognition accuracy. Real-time performance makes their method suitable for embedded AI hardware in dynamic UAV situations.

8.10. UAVid

The UAVid dataset has been extensively utilized as a benchmark by several researchers in the development of innovative methods for semantic segmentation in urban environments. Wang et al. [97] introduced the Bilateral Awareness Network (BANet) and applied it to the UAVid dataset, achieving a notable mean Intersection-over-Union (mIoU) score of 64.6%. BANet's ability to accurately segment various classes within high-resolution urban scenes was demonstrated through both quantitative metrics and qualitative analysis, outperforming other state-of-the-art models like the MSD benchmark.

Similarly, Rui Li et al. [98] proposed the Attention Aggregation Feature Pyramid Network (A^2-FPN) and reported significant improvements on the UAVid dataset. A^2-FPN achieved the highest mIoU across five out of eight classes, surpassing BANet by 1% in overall performance. The model's effectiveness was particularly evident in its ability to correctly identify moving vehicles, a challenging task for many segmentation models.

Libo Wang et al. [99] introduced UNetFormer, which further pushed the boundaries of semantic segmentation on the UAVid dataset. Achieving an impressive mIoU of 67.8%, UNetFormer outperformed several advanced networks, including ABCNet [114] and hybrid Transformer-based models like BANet and BoTNet [115]. UNetFormer demonstrated a strong ability to handle complex segmentation tasks, particularly in accurately identifying small objects like humans.

Lastly, Michael Ying Yang et al. [100] applied the Context Aggregation Network(CAN) to the UAVid dataset, achieving an mIoU score of 63.5% while maintaining a high processing speed of 15 frames per second (FPS). This model was noted for its ability to maintain consistency in both local and global scene semantics, making it a competitive choice for real-time applications in urban environments.

8.11. VERI-Wild

The VERI-Wild dataset has been extensively utilized as a benchmark by several researchers in the development of innovative methods for vehicle re-identification (ReID) in real-world scenarios. Meng et al. [102] introduced the Parsing-based View-aware Embedding Network (PVEN) and applied it to the VERI-Wild dataset, achieving significant improvements in mean average precision (mAP) across small, medium, and large test datasets, with increases of 47.4%, 47.2%, and 46.9%, respectively. PVEN's ability to perform view-aware feature alignment allowed it to consistently outperform state-of-the-art models, particularly in cumulative match characteristic (CMC) metrics, where it showed a 32.7% improvement over FDA-Net at rank 1.

Similarly, Lingxiao He et al. [103] evaluated the FastReID toolbox on the VERI-Wild dataset, highlighting its effectiveness in accurately identifying vehicles across various

conditions. FastReID achieved state-of-the-art performance, particularly in Rank-1 accuracy (R1-Accuracy) and mAP, showcasing its robustness in handling the complexities of vehicle ReID tasks in surveillance and traffic monitoring environments.

Fei Shen et al. [105] applied the GiT method on the VeRi-Wild dataset, securing top performance across all testing subsets, including Test3000 (T3000), Test5000 (T5000), and Test1000 (T1000). The GiT method outperformed the second-place method, PCRNet, by 0.41% in Rank-1 identification rate and 0.45% in mAP on the Test1000 subset. The study emphasized the importance of leveraging both global and local features, as GiT demonstrated superior generalization across different datasets and conditions. In a separate study, Fei Shen et al. [107] developed the Hybrid Pyramidal Graph Network (HPGN) approach, which achieved the highest Rank-1 identification rate among the evaluated methods on the VERI-Wild dataset, thus making more contributions to the advancing field of vehicle ReID. The findings highlighted the resilience of HPGN, especially in difficult circumstances such as fluctuating day and night situations, where alternative approaches exhibited a decrease in effectiveness.

Lastly, Khorramshahi et al. [109] presented a residual generation model that improved mAP by 2.0% and CMC1 by 1.0% compared to baseline models. The model's reliance on residual information, as indicated by a high alpha value ($\alpha = 0.94$), proved crucial in extracting robust features from the dataset. This self-supervised method further proved its adaptability and usefulness in vehicle ReID tasks by showcasing its efficacy on the VERI-Wild dataset.

9. Conclusions

In this survey paper, we looked at the current state of UAV datasets, highlighting their various applications, inherent challenges, and future directions. UAV datasets are essential in areas such as disaster management, surveillance, agriculture, environmental monitoring, and human behavior analysis. Advanced machine learning techniques have improved UAV capabilities, enabling more precise data collection and analysis. Despite their potential, UAV datasets face several challenges, including data quality, consistency, and the need for standardized annotation protocols. Ethical and privacy concerns necessitate strong frameworks to ensure responsible use. Future research should increase dataset diversity, integrate multimodal data, and improve real-time data processing. Improving annotation quality and promoting collaborative research and open data initiatives will increase dataset utility. To summarize, UAV datasets are at a critical stage of development, with significant opportunities for technological advancements. Addressing current challenges and focusing on future research directions will result in new discoveries, keeping UAV technology innovative and practical.

Author Contributions: Conceptualization, K.D.G. and M.M.R.; methodology, M.M.R.; writing—original draft preparation, M.M.R. and S.S.; writing—review and editing, M.K. and R.H.R.; supervision, K.D.G.; project administration, M.M.R., S.S., M.K., R.H.R., and K.D.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Georgia Tech HBCU Research Collaboration Forum (RCF) funding initiative. The funding provided by Georgia Tech has been instrumental in fostering collaborative research between Clark Atlanta University (CAU) and Georgia Tech. The findings, interpretations, and opinions expressed in this work are those of the authors and do not necessarily reflect the views or policies of Georgia Tech.

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

Appendix A. Visual Representation of Reviewed Datasets

The following images were captured from the papers in which they were presented as a new dataset or from the dataset repositories referenced in their paper where they were made available as public dataset repositories.

Appendix A.1. AIDER

Figure A1. Aerial Image Dataset for Applications in Emergency Response (AIDER): A selection of pictures from the augmented database.

Appendix A.2. BioDrone

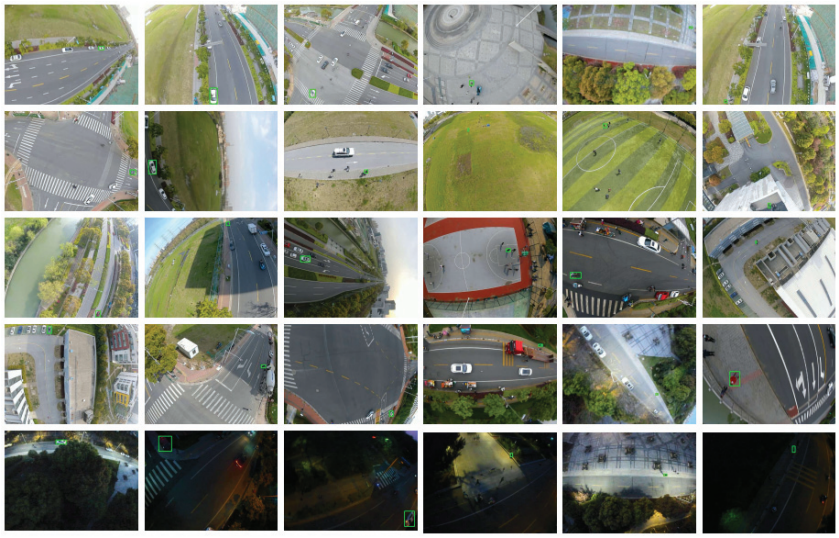


Figure A2. Illustrations of the flapping-wing UAV used for data collection and the representative data of BioDrone. Different flight attitudes for various scenes under three lighting conditions are included in the data acquisition process, ensuring that BioDrone can fully reflect the robust visual challenges of the flapping-wing UAVs.

Appendix A.3. ERA

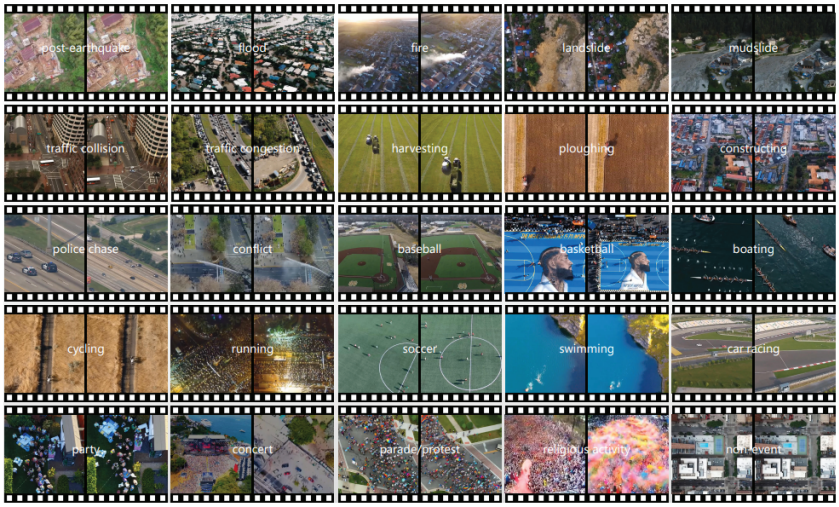
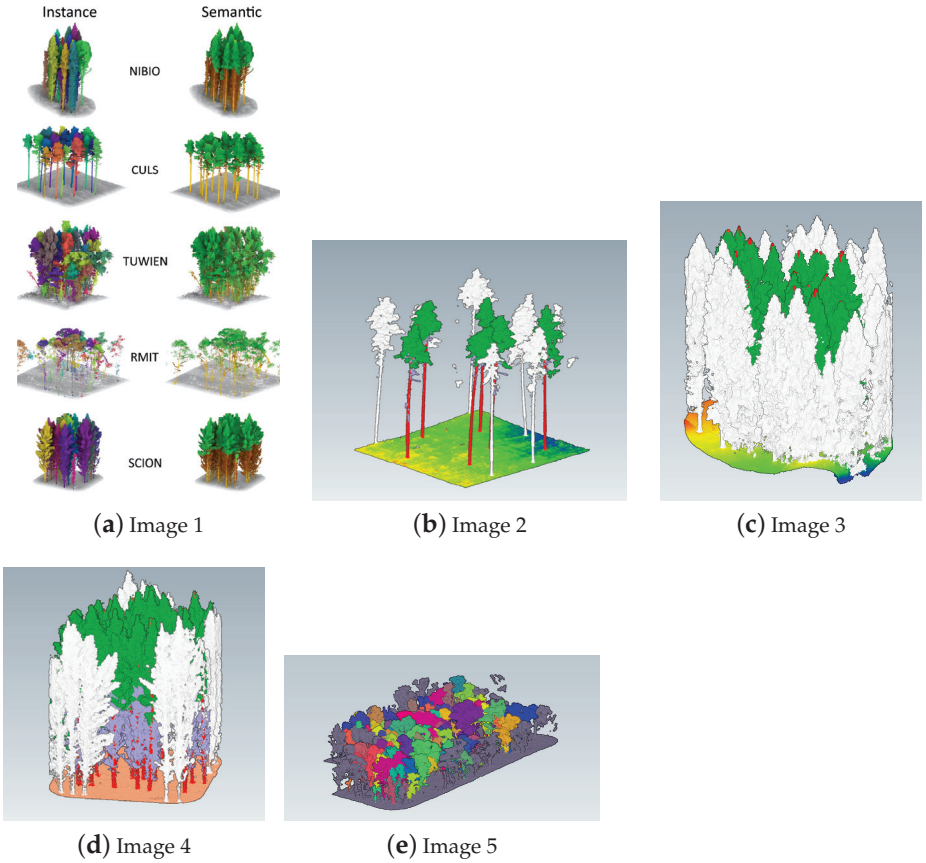


Figure A3. Overview of the ERA dataset. Overall, they have collected 2864 labeled video snippets for 24 event classes and 1 normal class: post-earthquake, flood, fire, landslide, mudslide, traffic collision, traffic congestion, harvesting, ploughing, constructing, police chase, conflict, baseball, basketball, boating, cycling, running, soccer, swimming, car racing, party, concert, parade/protest, religious activity, and non-event. For each class, we show the first (left) and last (right) frames of a video. Best viewed zoomed in color.

Appendix A.4. FOR-Instance



(**a**) Image 1 (**b**) Image 2 (**c**) Image 3

(**d**) Image 4 (**e**) Image 5

Figure A4. Samples of the various FOR-instance data collections' instance and semantic annotations.

Appendix A.5. UAVDark135

Figure A5. The first frames of representative scenes in newly constructed UAVDark135. Here, target ground-truths are marked out by green boxes and sequence names are located at the top left corner of the images. Dark special challenges like objects' unreliable color feature and objects' merging into the dark can be seen clearly.

Appendix A.6. UAV-Human

Figure A6. Examples of action videos in UAV-Human dataset. The first and second rows show two video sequences of significant camera motions and view variations, caused by continuously varying flight attitudes, speeds, and heights. The last three rows display action samples of the dataset, showing the diversities, e.g., distinct views, various capture sites, weathers, scales, and motion blur.

Appendix A.7. UAVid

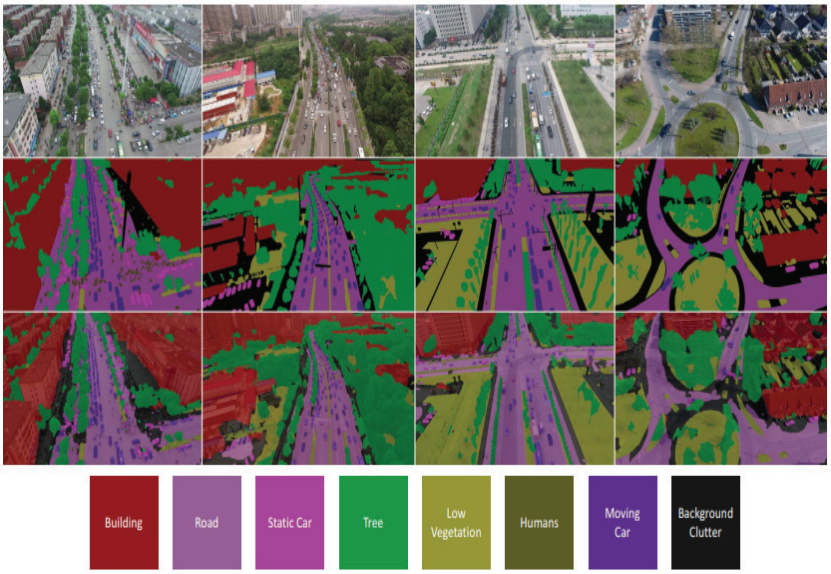


Figure A7. Example images and labels from UAVid dataset. First row shows the images captured by UAV. Second row shows the corresponding ground truth labels. Third row shows the prediction results of MS-Dilation net+PRT+FSO model. The last row shows the labels.

Appendix A.8. DarkTrack2021

Figure A8. Initial frames of specific sequences from the DarkTrack2021 archive. Objects being tracked are indicated by green boxes, and sequence names are shown in the top left corner of the photos.

Appendix A.9. VRAI

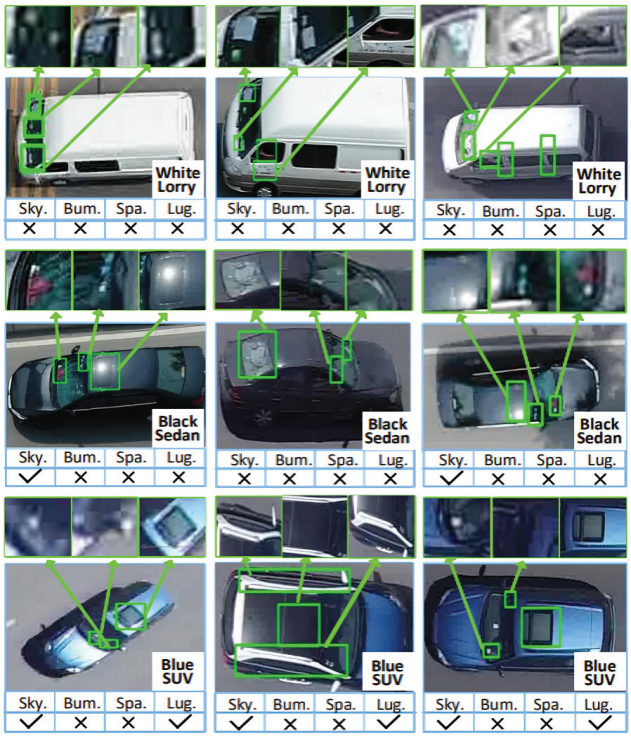


Figure A9. Illustration of a UAV-based vehicle Re-Identification (ReID) dataset displaying annotated distinguishing elements, such as skylights, bumpers, spare tires, and baggage racks, across diverse vehicle categories (e.g., white truck, black sedan, blue SUV). The green bounding boxes and arrows delineate particular vehicle components essential for identification, while variations in perspective and resolution underscore the difficulties of ReID from UAV imagery. The tabular annotations beneath each vehicle indicate the presence or absence of key elements for ReID.

Appendix A.10. VERI-Wild

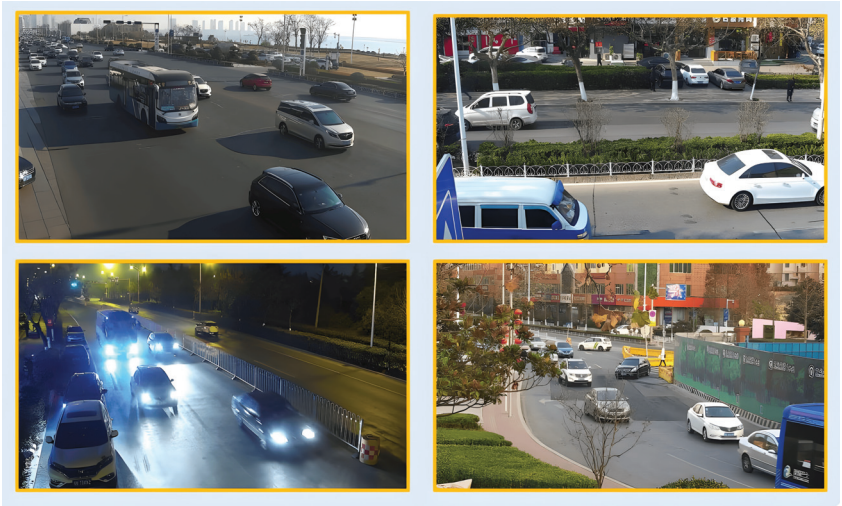


Figure A10. Exemplary photos extracted from the dataset. The dataset is obtained from a comprehensive real video surveillance system including 174 cameras strategically placed around an urban area spanning over 200 square kilometers.

Appendix A.11. RescueNet

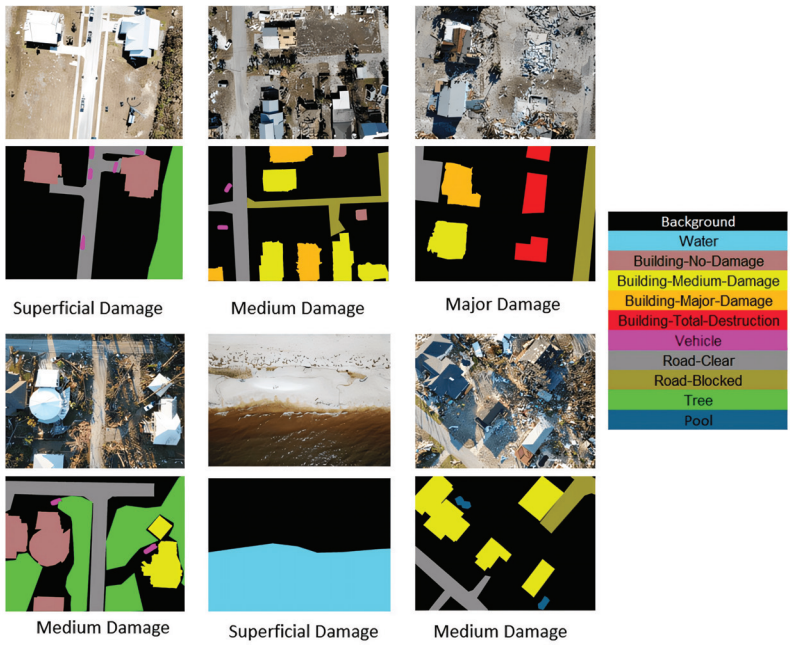


Figure A11. Graphical representation of complex scenes from the RescueNet dataset. The first and third rows display the original photos, while the lower rows provide the associated annotations for both semantic segmentation and image classification functions. Displayed on the right are the 10 classes, each represented by their segmentation color.

Appendix A.12. UAV-Assistant

Figure A12. This diagram illustrates the many modalities present in the UAV-Assistant dataset, which consists of a randomly chosen collection of images. The uppermost row displays color photos, the second row displays depth, the third row displays the normal map, and the last row displays flight silhouettes of the drone.

Appendix A.13. AU−AIR

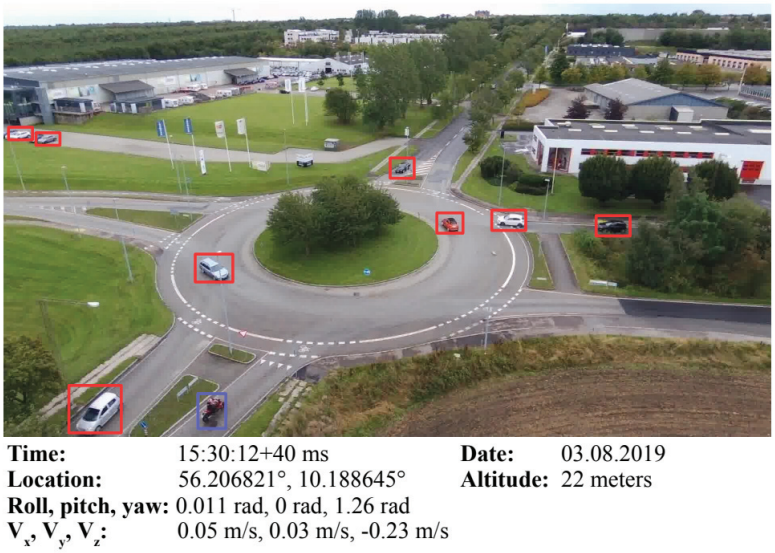


Figure A13. The AU−AIR dataset includes extracted frames that are annotated with object information, time stamp, current location, altitude, velocity of the UAV, and rotation data observed from the IMU sensor. This figure presents an exemplar of it.

Appendix A.14. UAV-Gesture

Figure A14. This diagram displays thirteen explicitly chosen gestures, each accompanied by a single picked image. Directions of hand movement are shown by the arrows. The amber color marks serve as approximate indicators of the initial and final locations of the palm for ONE iteration. Neither the Hover nor Land gestures are dynamic gestures.

Appendix A.15. Kite

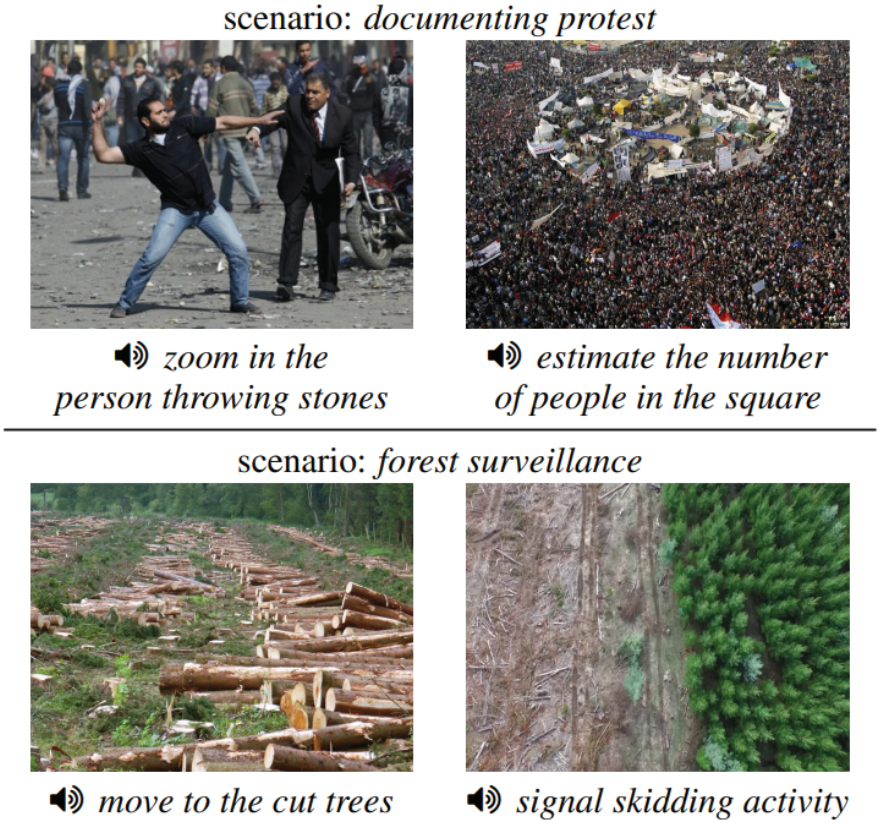


Figure A15. Exemplary commands and visual representations derived from the KITE dataset.

References

1. Mohsan, S.A.H.; Khan, M.A.; Noor, F.; Ullah, I.; Alsharif, M.H. Towards the unmanned aerial vehicles (UAVs): A comprehensive review. *Drones* **2022**, *6*, 147. [CrossRef]
2. Hu, S.; Zhao, X.; Huang, L.; Huang, K. Global instance tracking: Locating target more like humans. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 576–592. [CrossRef] [PubMed]
3. Duangsuwan, S.; Prapruetdee, P.; Subongkod, M.; Klubsuwan, K. 3D AQI Mapping Data Assessment of Low-Altitude Drone Real-Time Air Pollution Monitoring. *Drones* **2022**, *6*, 191. [CrossRef]
4. Meimetis, D.; Papaioannou, S.; Katsoni, P.; Lappas, V. An Architecture for Early Wildfire Detection and Spread Estimation Using Unmanned Aerial Vehicles, Base Stations, and Space Assets. *Drones Auton. Veh.* **2024**, *1*, 10006. [CrossRef]
5. Liu, Y.; Nie, J.; Li, X.; Ahmed, S.H.; Lim, W.Y.B.; Miao, C. Federated Learning in the Sky: Aerial-Ground Air Quality Sensing Framework With UAV Swarms. *IEEE Internet Things J.* **2021**, *8*, 9827–9837. [CrossRef]
6. Rahnemoonfar, M.; Chowdhury, T.; Murphy, R. RescueNet: A high resolution UAV semantic segmentation dataset for natural disaster damage assessment. *Sci. Data* **2023**, *10*, 913. [CrossRef]
7. Li, T.; Liu, J.; Zhang, W.; Ni, Y.; Wang, W.; Li, Z. Uav-human: A large benchmark for human behavior understanding with unmanned aerial vehicles. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16266–16275.
8. Kyrkou, C.; Theocharides, T. Deep-Learning-Based Aerial Image Classification for Emergency Response Applications Using Unmanned Aerial Vehicles. In Proceedings of the CVPR Workshops, Long Beach, CA, USA, 16–17 June 2019; pp. 517–525.
9. Bozcan, I.; Kayacan, E. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 8504–8510.
10. Mou, L.; Hua, Y.; Jin, P.; Zhu, X.X. Era: A data set and deep learning benchmark for event recognition in aerial videos [software and data sets]. *IEEE Geosci. Remote Sens. Mag.* **2020**, *8*, 125–133. [CrossRef]
11. Lyu, Y.; Vosselman, G.; Xia, G.S.; Yilmaz, A.; Yang, M.Y. UAVid: A semantic segmentation dataset for UAV imagery. *ISPRS J. Photogramm. Remote Sens.* **2020**, *165*, 108–119. [CrossRef]
12. Wang, P.; Jiao, B.; Yang, L.; Yang, Y.; Zhang, S.; Wei, W.; Zhang, Y. Vehicle re-identification in aerial imagery: Dataset and approach. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 460–469.

13. Puliti, S.; Pearse, G.; Surovỳ, P.; Wallace, L.; Hollaus, M.; Wielgosz, M.; Astrup, R. For-instance: A uav laser scanning benchmark dataset for semantic and instance segmentation of individual trees. *arXiv* **2023**, arXiv:2309.01279.
14. Lou, Y.; Bai, Y.; Liu, J.; Wang, S.; Duan, L. Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3235–3243.
15. Albanis, G.; Zioulis, N.; Dimou, A.; Zarpalas, D.; Daras, P. Dronepose: Photorealistic uav-assistant dataset synthesis for 3d pose estimation via a smooth silhouette loss. In Proceedings of the Computer Vision–ECCV 2020 Workshops, Glasgow, UK, 23–28 August 2020; Proceedings, Part II 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 663–681.
16. Oneata, D.; Cucu, H. Kite: Automatic speech recognition for unmanned aerial vehicles. *arXiv* **2019**, arXiv:1907.01195.
17. Perera, A.G.; Wei Law, Y.; Chahl, J. UAV-GESTURE: A dataset for UAV control and gesture recognition. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
18. Li, B.; Fu, C.; Ding, F.; Ye, J.; Lin, F. All-day object tracking for unmanned aerial vehicle. *IEEE Trans. Mob. Comput.* **2022**, *22*, 4515–4529. [CrossRef]
19. Ye, J.; Fu, C.; Cao, Z.; An, S.; Zheng, G.; Li, B. Tracker meets night: A transformer enhancer for UAV tracking. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3866–3873. [CrossRef]
20. Zhao, X.; Hu, S.; Wang, Y.; Zhang, J.; Hu, Y.; Liu, R.; Ling, H.; Li, Y.; Li, R.; Liu, K.; et al. Biodrone: A bionic drone-based single object tracking benchmark for robust vision. *Int. J. Comput. Vis.* **2024**, *132*, 1659–1684. [CrossRef]
21. Oktay, O.; Schlemper, J.; Folgoc, L.L.; Lee, M.; Heinrich, M.; Misawa, K.; Mori, K.; McDonagh, S.; Hammerla, N.Y.; Kainz, B.; et al. Attention u-net: Learning where to look for the pancreas. *arXiv* **2018**, arXiv:1804.03999.
22. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890.
23. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
24. Rahnemoonfar, M.; Chowdhury, T.; Sarkar, A.; Varshney, D.; Yari, M.; Murphy, R.R. Floodnet: A high resolution aerial imagery dataset for post flood scene understanding. *IEEE Access* **2021**, *9*, 89644–89654. [CrossRef]
25. Cheng, B.; Xiao, B.; Wang, J.; Shi, H.; Huang, T.S.; Zhang, L. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5386–5395.
26. Fang, H.S.; Xie, S.; Tai, Y.W.; Lu, C. Rmpe: Regional multi-person pose estimation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2334–2343.
27. Srisamosorn, V.; Kuwahara, N.; Yamashita, A.; Ogata, T.; Shirafuji, S.; Ota, J. Human position and head direction tracking in fisheye camera using randomized ferns and fisheye histograms of oriented gradients. *Vis. Comput.* **2020**, *36*, 1443–1456. [CrossRef]
28. Delibasis, K.K.; Plagianakos, V.P.; Maglogiannis, I. Pose recognition in indoor environments using a fisheye camera and a parametric human model. In Proceedings of the 2014 International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal, 5–8 January 2014; Volume 2, pp. 470–477.
29. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
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. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
32. Gilbert, A.; Trumble, M.; Malleson, C.; Hilton, A.; Collomosse, J. Fusing visual and inertial sensors with semantics for 3d human pose estimation. *Int. J. Comput. Vis.* **2019**, *127*, 381–397. [CrossRef]
33. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
34. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
35. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
36. Everingham, M.; Eslami, S.A.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
37. Kundu, A.; Vineet, V.; Koltun, V. Feature space optimization for semantic video segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3168–3175.
38. Chang, J.Y.; Lee, K.M. 2D–3D pose consistency-based conditional random fields for 3d human pose estimation. *Comput. Vis. Image Underst.* **2018**, *169*, 52–61. [CrossRef]
39. Nguyen, A.; Le, B. 3D point cloud segmentation: A survey. In Proceedings of the 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), Manila, Philippines, 12–15 November 2013; pp. 225–230.

40. Wei, X.S.; Zhang, C.L.; Liu, L.; Shen, C.; Wu, J. Coarse-to-fine: A RNN-based hierarchical attention model for vehicle re-identification. In Proceedings of the Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, 2–6 December 2018; Revised Selected Papers, Part II 14; Springer: Berlin/Heidelberg, Germany, 2019; pp. 575–591.
41. Liu, X.; Zhang, S.; Huang, Q.; Gao, W. Ram: A region-aware deep model for vehicle re-identification. In Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, USA, 23–27 July 2018; pp. 1–6.
42. Peddinti, V.; Povey, D.; Khudanpur, S. A time delay neural network architecture for efficient modeling of long temporal contexts. In Proceedings of the Interspeech, Dresden, Germany, 6–10 September 2015; pp. 3214–3218.
43. Povey, D.; Ghoshal, A.; Boulianne, G.; Burget, L.; Glembek, O.; Goel, N.; Hannemann, M.; Motlicek, P.; Qian, Y.; Schwarz, P.; et al. The Kaldi speech recognition toolkit. In Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, Waikoloa, HI, USA, 11–15 December 2011.
44. Vondrick, C.; Patterson, D.; Ramanan, D. Efficiently scaling up crowdsourced video annotation: A set of best practices for high quality, economical video labeling. *Int. J. Comput. Vis.* **2013**, *101*, 184–204. [CrossRef]
45. Chéron, G.; Laptev, I.; Schmid, C. P-cnn: Pose-based cnn features for action recognition. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 3218–3226.
46. Mueller, M.; Smith, N.; Ghanem, B. A benchmark and simulator for UAV tracking. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 445–461.
47. Li, S.; Yeung, D.Y. Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
48. Mayer, C.; Danelljan, M.; Paudel, D.P.; Van Gool, L. Learning target candidate association to keep track of what not to track. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 13444–13454.
49. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 583–596. [CrossRef]
50. Dimensions AI: The Most Advanced Scientific Research Database. Available online: https://up42.com/blog/full-spectrum-multispectral-imagery-and-hyperspectral-imagery (accessed on 12 November 2024).
51. Strudel, R.; Garcia, R.; Laptev, I.; Schmid, C. Segmenter: Transformer for Semantic Segmentation. *arXiv* **2021**, arXiv:2105.05633.
52. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning Spatiotemporal Features with 3D Convolutional Networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4489–4497. [CrossRef]
53. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *arXiv* **2015**, arXiv:1411.4038.
54. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv* **2015**, arXiv:1505.04597.
55. Yang, L.; Luo, P.; Loy, C.C.; Tang, X. A Large-Scale Car Dataset for Fine-Grained Categorization and Verification. *arXiv* **2015**, arXiv:1506.08959.
56. Schroff, F.; Kalenichenko, D.; Philbin, J. FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 815–823. [CrossRef]
57. Liu, X.; Liu, W.; Mei, T.; Ma, H. A Deep Learning-Based Approach to Progressive Vehicle Re-identification for Urban Surveillance. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Volume 9906, pp. 869–884. [CrossRef]
58. Liu, H.; Tian, Y.; Wang, Y.; Pang, L.; Huang, T. Deep Relative Distance Learning: Tell the Difference between Similar Vehicles. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2167–2175. [CrossRef]
59. Yuan, Y.; Yang, K.; Zhang, C. Hard-Aware Deeply Cascaded Embedding. *arXiv* **2017**, arXiv:1611.05720.
60. Zheng, Z.; Zheng, L.; Yang, Y. Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline in vitro. *arXiv* **2017**, arXiv:1701.07717.
61. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv* **2020**, arXiv:1703.10593.
62. Tekin, B.; Sinha, S.N.; Fua, P. Real-Time Seamless Single Shot 6D Object Pose Prediction. *arXiv* **2018**, arXiv:1711.08848.
63. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. *arXiv* **2019**, arXiv:1902.09630.
64. O'shea, K.; Nash, R. An introduction to convolutional neural networks. *arXiv* **2015**, arXiv:1511.08458.
65. 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.
66. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710.
67. Sy Nguyen, V.; Jung, J.; Jung, S.; Joe, S.; Kim, B. Deployable Hook Retrieval System for UAV Rescue and Delivery. *IEEE Access* **2021**, *9*, 74632–74645. [CrossRef]

68. Zhao, J.; Zhang, X.; Gao, C.; Qiu, X.; Tian, Y.; Zhu, Y.; Cao, W. Rapid Mosaicking of Unmanned Aerial Vehicle (UAV) Images for Crop Growth Monitoring Using the SIFT Algorithm. *Remote Sens.* **2019**, *11*, 1226. [CrossRef]
69. Yu, J.; Gao, H.; Zhou, D.; Liu, J.; Gao, Q.; Ju, Z. Deep Temporal Model-Based Identity-Aware Hand Detection for Space Human–Robot Interaction. *IEEE Trans. Cybern.* **2022**, *52*, 13738–13751. [CrossRef]
70. Saeed, Z.; Yousaf, M.H.; Ahmed, R.; Velastin, S.A.; Viriri, S. On-board small-scale object detection for unmanned aerial vehicles (UAVs). *Drones* **2023**, *7*, 310. [CrossRef]
71. Walambe, R.; Marathe, A.; Kotecha, K. Multiscale object detection from drone imagery using ensemble transfer learning. *Drones* **2021**, *5*, 66. [CrossRef]
72. Gupta, H.; Verma, O.P. Monitoring and surveillance of urban road traffic using low altitude drone images: A deep learning approach. *Multimed. Tools Appl.* **2022**, *81*, 19683–19703. [CrossRef]
73. Bountos, N.I.; Ouaknine, A.; Rolnick, D. FoMo-Bench: A multi-modal, multi-scale and multi-task Forest Monitoring Benchmark for remote sensing foundation models. *arXiv* **2024**, arXiv:2312.10114.
74. Zhang, C.; Song, C.; Zaforemska, A.; Zhang, J.; Gaulton, R.; Dai, W.; Xiao, W. Individual tree segmentation from UAS Lidar data based on hierarchical filtering and clustering. *Int. J. Digit. Earth* **2024**, *17*, 2356124. [CrossRef]
75. Albanis, G.N.; Zioulis, N.; Chatzitofis, A.; Dimou, A.; Zarpalas, D.; Daras, P. On end-to-end 6DOF object pose estimation and robustness to object scale. In Proceedings of the ML Reproducibility Challenge, online, 23–28 August 2020.
76. Chen, B.; Parra, A.; Cao, J.; Li, N.; Chin, T.J. End-to-end learnable geometric vision by backpropagating pnp optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8100–8109.
77. Sun, K.; Xiao, B.; Liu, D.; Wang, J. Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5693–5703.
78. Kyrkou, C.; Theocharides, T. EmergencyNet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 1687–1699. [CrossRef]
79. Alrayes, F.S.; Alotaibi, S.S.; Alissa, K.A.; Maashi, M.; Alhogail, A.; Alotaibi, N.; Mohsen, H.; Motwakel, A. Artificial intelligence-based secure communication and classification for drone-enabled emergency monitoring systems. *Drones* **2022**, *6*, 222. [CrossRef]
80. Fu, C.; Yao, L.; Zuo, H.; Zheng, G.; Pan, J. SAM-DA: UAV Tracks Anything at Night with SAM-Powered Domain Adaptation. *arXiv* **2024**, arXiv:2307.01024.
81. Fu, C.; Lu, K.; Zheng, G.; Ye, J.; Cao, Z.; Li, B.; Lu, G. Siamese Object Tracking for Unmanned Aerial Vehicle: A Review and Comprehensive Analysis. *arXiv* **2022**, arXiv:2205.04281. [CrossRef]
82. Azmat, U.; Alotaibi, S.S.; Mudawi, N.A.; Alabduallah, B.I.; Alonazi, M.; Jalal, A.; Park, J. An Elliptical Modeling Supported System for Human Action Deep Recognition Over Aerial Surveillance. *IEEE Access* **2023**, *11*, 75671–75685. [CrossRef]
83. Lin, W.; Karlinsky, L.; Shvetsova, N.; Possegger, H.; Kozinski, M.; Panda, R.; Feris, R.; Kuehne, H.; Bischof, H. MAtch, eXpand and Improve: Unsupervised Finetuning for Zero-Shot Action Recognition with Language Knowledge. *arXiv* **2023**, arXiv:2303.08914.
84. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. *arXiv* **2021**, arXiv:2103.00020.
85. Rasheed, H.; Khattak, M.U.; Maaz, M.; Khan, S.; Khan, F.S. Fine-tuned CLIP Models are Efficient Video Learners. *arXiv* **2023**, arXiv:2212.03640.
86. Huang, Z.; Qin, Y.; Lin, X.; Liu, T.; Feng, Z.; Liu, Y. Motion-Driven Spatial and Temporal Adaptive High-Resolution Graph Convolutional Networks for Skeleton-Based Action Recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 1868–1883. [CrossRef]
87. Hu, Z.; Pan, Z.; Wang, Q.; Yu, L.; Fei, S. Forward-reverse adaptive graph convolutional networks for skeleton-based action recognition. *Neurocomput.* **2022**, *492*, 624–636. [CrossRef]
88. Zhu, J.; Tang, H.; Cheng, Z.Q.; He, J.Y.; Luo, B.; Qiu, S.; Li, S.; Lu, H. DCPT: Darkness Clue-Prompted Tracking in Nighttime UAVs. *arXiv* **2024**, arXiv:2309.10491.
89. Bhat, G.; Danelljan, M.; Gool, L.V.; Timofte, R. Learning discriminative model prediction for tracking. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6182–6191.
90. Ye, J.; Fu, C.; Zheng, G.; Cao, Z.; Li, B. DarkLighter: Light Up the Darkness for UAV Tracking. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 3079–3085. [CrossRef]
91. Fu, C.; Cao, Z.; Li, Y.; Ye, J.; Feng, C. Siamese Anchor Proposal Network for High-Speed Aerial Tracking. *arXiv* **2021**, arXiv:2012.10706.
92. Ferdous, S.N.; Li, X.; Lyu, S. Uncertainty aware multitask pyramid vision transformer for uav-based object re-identification. In Proceedings of the 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 16–19 October 2022; pp. 2381–2385.
93. Chen, S.; Ye, M.; Du, B. Rotation invariant transformer for recognizing object in uavs. In Proceedings of the 30th ACM International Conference on Multimedia, Lisboa, Portugal, 10–14 October 2022; pp. 2565–2574.
94. Azmat, U.; Alotaibi, S.S.; Abdelhaq, M.; Alsufyani, N.; Shorfuzzaman, M.; Jalal, A.; Park, J. Aerial Insights: Deep Learning-Based Human Action Recognition in Drone Imagery. *IEEE Access* **2023**, *11*, 83946–83961. [CrossRef]

95. Papaioannidis, C.; Makrygiannis, D.; Mademlis, I.; Pitas, I. Learning Fast and Robust Gesture Recognition. In Proceedings of the 2021 29th European Signal Processing Conference (EUSIPCO), Dublin, Ireland, 23–27 August 2021; pp. 761–765. [CrossRef]
96. Yang, F.; Sakti, S.; Wu, Y.; Nakamura, S. Make Skeleton-based Action Recognition Model Smaller, Faster and Better. *arXiv* **2020**, arXiv:1907.09658.
97. Wang, L.; Li, R.; Wang, D.; Duan, C.; Wang, T.; Meng, X. Transformer Meets Convolution: A Bilateral Awareness Network for Semantic Segmentation of Very Fine Resolution Urban Scene Images. *arXiv* **2022**, arXiv:2106.12413. [CrossRef]
98. Li, R.; Wang, L.; Zhang, C.; Duan, C.; Zheng, S. A2-FPN for semantic segmentation of fine-resolution remotely sensed images. *Int. J. Remote Sens.* **2022**, *43*, 1131–1155. [CrossRef]
99. Wang, L.; Li, R.; Zhang, C.; Fang, S.; Duan, C.; Meng, X.; Atkinson, P.M. UNetFormer: A UNet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery. *ISPRS J. Photogramm. Remote Sens.* **2022**, *190*, 196–214. [CrossRef]
100. Yang, M.Y.; Kumaar, S.; Lyu, Y.; Nex, F. Real-time semantic segmentation with context aggregation network. *ISPRS J. Photogramm. Remote Sens.* **2021**, *178*, 124–134. [CrossRef]
101. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 325–341.
102. Meng, D.; Li, L.; Liu, X.; Li, Y.; Yang, S.; Zha, Z.J.; Gao, X.; Wang, S.; Huang, Q. Parsing-based view-aware embedding network for vehicle re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 7103–7112.
103. He, L.; Liao, X.; Liu, W.; Liu, X.; Cheng, P.; Mei, T. Fastreid: A pytorch toolbox for general instance re-identification. In Proceedings of the 31st ACM International Conference on Multimedia, Ottawa, ON, Canada, 29 October–3 November 2023; pp. 9664–9667.
104. Alfasly, S.; Hu, Y.; Li, H.; Liang, T.; Jin, X.; Liu, B.; Zhao, Q. Multi-label-based similarity learning for vehicle re-identification. *IEEE Access* **2019**, *7*, 162605–162616. [CrossRef]
105. Shen, F.; Xie, Y.; Zhu, J.; Zhu, X.; Zeng, H. Git: Graph interactive transformer for vehicle re-identification. *IEEE Trans. Image Process.* **2023**, *32*, 1039–1051. [CrossRef]
106. Liu, X.; Liu, W.; Zheng, J.; Yan, C.; Mei, T. Beyond the parts: Learning multi-view cross-part correlation for vehicle re-identification. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 907–915.
107. Shen, F.; Zhu, J.; Zhu, X.; Xie, Y.; Huang, J. Exploring spatial significance via hybrid pyramidal graph network for vehicle re-identification. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 8793–8804. [CrossRef]
108. Kuma, R.; Weill, E.; Aghdasi, F.; Sriram, P. Vehicle re-identification: An efficient baseline using triplet embedding. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–9.
109. Khorramshahi, P.; Peri, N.; Chen, J.c.; Chellappa, R. The devil is in the details: Self-supervised attention for vehicle re-identification. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part XIV 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 369–386.
110. Luo, H.; Gu, Y.; Liao, X.; Lai, S.; Jiang, W. Bag of tricks and a strong baseline for deep person re-identification. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019.
111. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
112. Ye, J.; Fu, C.; Zheng, G.; Paudel, D.P.; Chen, G. Unsupervised Domain Adaptation for Nighttime Aerial Tracking. *arXiv* **2022**, arXiv:2203.10541.
113. Cao, Z.; Fu, C.; Ye, J.; Li, B.; Li, Y. SiamAPN++: Siamese Attentional Aggregation Network for Real-Time UAV Tracking. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 3086–3092. [CrossRef]
114. Li, R.; Zheng, S.; Zhang, C.; Duan, C.; Wang, L.; Atkinson, P.M. ABCNet: Attentive bilateral contextual network for efficient semantic segmentation of Fine-Resolution remotely sensed imagery. *ISPRS J. Photogramm. Remote Sens.* **2021**, *181*, 84–98. [CrossRef]
115. Srinivas, A.; Lin, T.Y.; Parmar, N.; Shlens, J.; Abbeel, P.; Vaswani, A. Bottleneck Transformers for Visual Recognition. *arXiv* **2021**, arXiv:2101.11605.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

MDPI AG
Grosspeteranlage 5
4052 Basel
Switzerland
Tel.: +41 61 683 77 34

Algorithms Editorial Office
E-mail: algorithms@mdpi.com
www.mdpi.com/journal/algorithms

Disclaimer/Publisher's Note: The title and front matter of this reprint are at the discretion of the Guest Editors. The publisher is not responsible for their content or any associated concerns. The statements, opinions and data contained in all individual articles are solely those of the individual Editors and contributors and not of MDPI. MDPI disclaims responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Academic Open
Access Publishing

mdpi.com

ISBN 978-3-7258-6351-8