*future internet*

# Innovative Topologies and Algorithms for Neural Networks

Edited by

Salvatore Graziani and Maria Gabriella Xibilia

Printed Edition of the Special Issue Published in *Future Internet*

www.mdpi.com/journal/futureinternet

MDPI

# Innovative Topologies and Algorithms for Neural Networks

# Innovative Topologies and Algorithms for Neural Networks

Editors

**Salvatore Graziani**
**Maria Gabriella Xibilia**

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

*Editors*

| | |
|---|---|
| Salvatore Graziani | Maria Gabriella Xibilia |
| University of Catania | University of Messina |
| Italy | Italy |

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

This is a reprint of articles from the Special Issue published online in the open access journal *Future Internet* (ISSN 1999-5903) (available at: https://www.mdpi.com/journal/futureinternet/special_issues/Innovative_topologies_neural_networks).

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

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

**ISBN 978-3-0365-0284-7 (Hbk)**
**ISBN 978-3-0365-0285-4 (PDF)**

# Contents

# About the Editors

**Salvatore Graziani** received his M.S. in electronic engineering and Ph.D. in electrical engineering from the Università degli Studi di Catania, Italy, in 1990 and 1994, respectively. Since 1990, he has been with the Dipartimento di Ingegneria Elettrica, Elettronica e Informatica, Università di Catania, where he is an Associate Professor of electric and electronic measurement and instrumentation. His primary research interests lie in the field of sensors and actuators, as well as soft sensors. He has coauthored several scientific papers and books.

**Maria Gabriella Xibilia** received her M.S. degree in electronic engineering and Ph.D. in electrical engineering from the Università degli Studi di Catania, Catania, Italy, in 1991 and 1995, respectively. Since 1998, she has been with the Dipartimento di Ingegneria, Università di Messina, Messina, Italy, where she is currently an Associate Professor of automatic control. She has coauthored several scientific papers and books.

Her current research interests include system identification, soft sensors, process control, nonlinear systems, and machine learning.

# Preface to "Innovative Topologies and Algorithms for Neural Networks"

Interest in the study of deep neural networks in the field of neural computation has increased, both as it regards to new training procedures and topologies, as well as significant applications. In particular, greater attention is being paid to challenging applications that are not adequately addressed by classical machine learning methods. Consequently, the use of deep structures has significantly improved state-of-the-art applications in many fields, such as object and gesture recognition, speech and language processing, and the Internet of Things (IoT). This book is comprised of discussions and analyses of relevant applications in the fields of speech and text analysis, object and gesture recognition, medical applications, IoT implementations, and sentiment analysis. Successful solutions to complex problems, such as those examined in the contributions noted above, are closely linked to identifying suitable network architectures. In this book, long short-term memory (LSTM) and convolutional neural network (CNN)-derived architectures are the most commonly used neural structures. Furthermore, in many of the contributions, a deep interplay exists between the adopted neural structures and the investigated application, leading to the proposal of tailored architectures. The authors give significant contributions to the above-mentioned fields by merging theoretical aspects and relevant applications.

**Salvatore Graziani, Maria Gabriella Xibilia**
*Editors*

*future internet*

MDPI

*Editorial*

# Innovative Topologies and Algorithms for Neural Networks

**Salvatore Graziani [1] and Maria Gabriella Xibilia [2,*]**

[1] Dipartimento di Ingegneria Elettrica, Elettronica e Informatica, University of Catania, Viale Andrea Doria 6, 95125 Catania, Italy; salvatore.graziani@unict.it

[2] Dipartimento di Ingegneria, University of Messina, Contrada di Dio, S. Agata, 98166 Messina ME, Italy

* Correspondence: mxibilia@unime.it

Received: 30 June 2020; Accepted: 2 July 2020; Published: 11 July 2020

**Abstract:** The introduction of new topologies and training procedures to deep neural networks has solicited a renewed interest in the field of neural computation. The use of deep structures has significantly improved the state of the art in many applications, such as computer vision, speech and text processing, medical applications, and IoT (Internet of Things). The probability of a successful outcome from a neural network is linked to selection of an appropriate network architecture and training algorithm. Accordingly, much of the recent research on neural networks is devoted to the study and proposal of novel architectures, including solutions tailored to specific problems. The papers of this Special Issue make significant contributions to the above-mentioned fields by merging theoretical aspects and relevant applications. Twelve papers are collected in the issue, addressing many relevant aspects of the topic.

**Keywords:** autoencoders; long-short-term memory networks; convolution neural Networks; object recognition; sentiment analysis; text recognition; gesture recognition; IoT (Internet of Thing) systems; medical applications

## 1. Introduction

Interest in the study of deep neural networks in the field of neural computation is increasing, both as it regards new training procedures and topologies, and significant applications. In particular, greater attention is being paid to challenging applications, which are not adequately addressed by classical machine learning methods. Consequently, the use of deep structures has significantly improved the state of the art in many fields, such as object and gesture recognition, speech and language processing, and the Internet of Things (IoT).

This Special Issue comprises discussions and analysis of relevant applications in the fields of speech and text analysis; object and gesture recognition; medical applications; IoT implementations; and sentiment analysis.

Successful solutions to complex problems, such as those examined in the contributions noted above, are closely linked to the identification of suitable network architectures. In this issue, long short-term memory (LSTM)- and convolutional neural network (CNN)-derived architectures are the most commonly used neural structures.

Furthermore, in many of the contributions in this issue, a deep interplay exists between the adopted neural structure and the investigated application, leading to the proposal of tailored architectures.

The papers of this Special Issue make significant contributions to the above-mentioned fields by merging theoretical aspects and relevant applications. Nevertheless, topics related to the choice of neural structure and learning algorithm, network sizing, and selection of hyperparameters require further examination and are the focus of a vivid research interest.

## 2. Contributions

The papers included in this Special Issue of *Future Internet* provide interesting examinations of deep neural networks, considering both theoretical contributions and relevant applications. Case studies are reported for object recognition, text recognition and sentiment analysis, medical applications, and other emerging fields.

The first paper [1] investigates pedestrian attribute recognition within surveillance scenarios. This challenging task is approached as a form of multi-label classification. The authors propose a novel model based on a graph convolutional network (GCN), which uses a CNN to extract pedestrian features and a correlation matrix between labels to propagate information between nodes. Reported results show that the approach proposed by the authors outperforms other existing state-of-the-art methods.

The second paper [2] focuses on the exploding field of IoT systems. Fog computing is used to process the huge amount of data produced by IoT applications. In the paper presented here, Deep Neural Networks Partitioning for Constrained IoT Devices is proposed as a new algorithm to partition neural networks for efficient distributed execution. The authors show that the partitioning offered by popular machine learning frameworks, such as TensorFlow, or by the general-purpose framework METIS, may produce invalid partitioning for highly constrained systems, while the proposed algorithm can be more efficient.

In the third paper [3], text sentiment analysis is addressed as an important and challenging application. A sentiment-feature-enhanced deep neural network (SDNN) is proposed to integrate sentiment linguistic knowledge into a deep neural network via a sentiment attention mechanism. This helps to select the crucial sentiment-relevant context words by leveraging the sentiment lexicon in an attention mechanism, bridging the gap between traditional sentiment linguistic knowledge and deep learning methods. Experimental results are reported showing that the proposed structure achieved better performance than competitors for text sentiment classification tasks.

The fourth paper [4] deals with another relevant application field, i.e., gesture recognition in video. A neural network comprising an alternate fusion of a 3D CNN and ConvLSTM, called the Multiple extraction and Multiple prediction (MEMP) network, is proposed. The main feature of the MEMP network is the repeated extraction and prediction of the temporal and spatial feature information of gesture video, which enables a high accuracy rate to be obtained. The performance of the proposed method is tested on benchmark datasets, showing high accuracy.

A recognition problem is also the topic of [5]. Specifically, ship detection and recognition are addressed to better manage port resources. The authors propose an on-site processing approach, called Embedded Ship Detection and Recognition using Deep Learning (ESDR-DL). Processing of a video stream using embedded devices, and a two-stage neural network composed of a DNet for ship detection and a CNet for ship recognition, also running on embedded devices, is proposed. The ESDR-DL is deployed at the Dongying port of China, where it has been running for over a year.

A medical application is the subject of [6], in which the tooth-marked tongue, an important indicator in traditional Chinese medicinal diagnosis, is considered. This paper is an example of a typical application in which a correct diagnosis relies on the experience and knowledge of the practitioner. In the study, a visual explanation method uses a CNN to extract features and a Gradient-weighted Class Activation Mapping is used to produce a coarse localization map. Experimental results demonstrate the effectiveness of the proposed method.

Paper [7] concerns human activity recognition. The paper introduces a new framework combining 3D-CNN and LSTM networks. The framework integrates a motion map with the next video frame to obtain a new motion map by increasing the training video length iteratively. A linear weighted fusion scheme is then used to fuse the network feature maps into spatio-temporal features. Finally, an LSTM encoder-decoder is used for predictions. Public benchmark datasets are used to prove the effectiveness of the proposed method.

An LSTM-conditional random field model (LSTM-CRF model) with an integrity algorithm is proposed in [8]. The method incorporates the advantages of the data-driven method and dependency syntax, and improves the precision rate of the elements without undermining the recall rate. Cross-domain experiments based on a multi-industry corpus in the financial field are reported.

Object detection is addressed in [9], where feature fusion is added to an object detection network to obtain a better CNN feature, thus improving the performance of a small object. An attention mechanism is applied to an object detection network to enhance the impact of significant features and weaken background interference. Empirical evaluation on a public dataset demonstrates the effectiveness of the proposed approach.

Paper [10] uses a bidirectional LSTM model as an approach to address named entity recognition (NER) in natural language processing tasks in Arabic text. The LSTM network can process sequences and relate them to each part of the text, making it useful for NER tasks. Pre-trained word embedding is used to train the inputs that are fed into the LSTM network. The proposed model is evaluated on a popular dataset.

A medical application is, again, the topic of [11], where facial nerve paralysis (FNP) is considered. The use of objective measurements can reduce the frequency of errors caused by subjective methods. A single CNN, trained directly from images classified by neurologists, is proposed. The proposed CNN successfully matched the neurologists' classification.

Text classification returns in [12], in which the case of Chinese text is considered. After comparing different methods, LSTM and CNN approaches are selected as deep learning methods to classify Chinese text. Two layers of LSTM and one layer of CNN are integrated into a new model, labelled the BLSTM-C model. The LSTM is responsible for obtaining a sequence output based on past and future contexts, which is then input to the convolutional layer for feature extraction. The model exhibited remarkable performance in classification of Chinese texts.

**Funding:** This research received no external funding.

**Acknowledgments:** The guest editors wish to thank all the contributing authors, the professional reviewers for their precious help with the review assignments, and the excellent editorial support from the Future Internet journal at every stage of the publication process of this special issue.

**Conflicts of Interest:** Declare conflicts of interest or state "The authors declare no conflict of interest".

## References

1. Song, X.; Yang, H.; Zhou, C. Pedestrian Attribute Recognition with Graph Convolutional Network in Surveillance Scenarios. *Future Internet* **2019**, *11*, 245. [CrossRef]
2. de Oliveira, F.M.C.; Borin, E. Partitioning convolutional neural networks to maximize the inference rate on constrained IoT devices. *Future Internet* **2019**, *11*, 209. [CrossRef]
3. Li, W.; Liu, P.; Zhang, Q.; Liu, W. An improved approach for text sentiment classification based on a deep neural network via a sentiment attention mechanism. *Future Internet* **2019**, *11*, 96. [CrossRef]
4. Zhang, X.; Li, X. Dynamic gesture recognition based on MEMP network. *Future Internet* **2019**, *11*, 91. [CrossRef]
5. Zhao, H.; Zhang, W.; Sun, H.; Xue, B. Embedded deep learning for ship detection and recognition. *Future Internet* **2019**, *11*, 53. [CrossRef]
6. Sun, Y.; Dai, S.; Li, J.; Zhang, Y.; Li, X. Tooth-marked tongue recognition using gradient-weighted class activation maps. *Future Internet* **2019**, *11*, 45. [CrossRef]
7. Arif, S.; Wang, J.; Ul Hassan, T.; Fei, Z. 3D-CNN-based fused feature maps with LSTM applied to action recognition. *Future Internet* **2019**, *11*, 42. [CrossRef]
8. Xu, D.; Ge, R.; Niu, Z. Forward-looking element recognition based on the LSTM-CRF model with the integrity algorithm. *Future Internet* **2019**, *11*, 17. [CrossRef]
9. Zhang, Y.; Chen, Y.; Huang, C.; Gao, M. Object detection network based on feature fusion and attention mechanism. *Future Internet* **2019**, *11*, 9. [CrossRef]
10. Ali, M.N.A.; Tan, G.; Hussain, A. Bidirectional recurrent neural network approach for Arabic named entity recognition. *Future Internet* **2019**, *11*, 123. [CrossRef]

11. Song, A.; Wu, Z.; Ding, X.; Hu, Q.; Di, X. Neurologist standard classification of facial nerve paralysis with deep neural networks. *Future Internet* **2019**, *11*, 111. [CrossRef]
12. Li, Y.; Wang, X.; Xu, P. Chinese text classification model based on deep learning. *Future Internet* **2019**, *11*, 113. [CrossRef]

*future internet*

Article

# Pedestrian Attribute Recognition with Graph Convolutional Network in Surveillance Scenarios

**Xiangpeng Song *, Hongbin Yang and Congcong Zhou**

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; hbyoungshu@staff.shu.edu.cn (H.Y.); zhoucongcong@shu.edu.cn (C.Z.)
* Correspondence: sxptom@shu.edu.cn; Tel.: +86-18019118350

Received: 23 October 2019; Accepted: 15 November 2019; Published: 19 November 2019

**Abstract:** Pedestrian attribute recognition is to predict a set of attribute labels of the pedestrian from surveillance scenarios, which is a very challenging task for computer vision due to poor image quality, continual appearance variations, as well as diverse spatial distribution of imbalanced attributes. It is desirable to model the label dependencies between different attributes to improve the recognition performance as each pedestrian normally possesses many attributes. In this paper, we treat pedestrian attribute recognition as multi-label classification and propose a novel model based on the graph convolutional network (GCN). The model is mainly divided into two parts, we first use convolutional neural network (CNN) to extract pedestrian feature, which is a normal operation processing image in deep learning, then we transfer attribute labels to word embedding and construct a correlation matrix between labels to help GCN propagate information between nodes. This paper applies the object classifiers learned by GCN to the image representation extracted by CNN to enable the model to have the ability to be end-to-end trainable. Experiments on pedestrian attribute recognition dataset show that the approach obviously outperforms other existing state-of-the-art methods.

**Keywords:** pedestrian attribute recognition; graph convolutional network; multi-label learning

## 1. Introduction

Video surveillance is a part of our daily life, with the advent of the era of artificial intelligence, intelligent video analytics attaches great importance to the modern city since it can pre-alarm abnormal behaviors or events [1]. In this paper, we mainly focus on the pedestrian in the surveillance system. Human attribute analysis has recently drawn a remarkable amount of attentions by researchers for person detection and re-identification, as well as widely applied in plenty of aspects [2]; besides, pedestrian structured representation can obviously reduce surveillance video storage and improve pedestrian retrieve speed in the surveillance system.

However, there are still plenty of challenges. For one thing, human attributes naturally involve largely imbalanced data distribution. For example, when collecting the attribute “Bald”, most of them will be labeled as “No Bald” and its imbalanced ratio to the “Bald” class is usually very large [3]. For another, there are plenty of uncertainties in practical scenarios, the resolution of images may be lower and the human body may be blocked by other things [4]. Furthermore, the collecting of labeled samples is labor-consuming.

Extreme learning machine (ELM) has gained increasing interest from various research fields for certain years. Apart from classification and regression, ELM has recently been extended for clustering, feature selection, representational learning, and many other learning tasks [5]. As an efficient single-hidden-layer feed forward neural network, which has generally good performance and fast learning speed, ELM has been applied in a variety of domains, such as computer vision, energy disaggregation [6], and speech enhancement [7]. In [8], authors proposed a novel pedestrian detection method using multimodal Histogram of Oriented Gradient for pedestrian feature extraction and extreme learning machine for classification to reduce the detection rate of false positives and accelerate processing speed. The experimental results have proved the efficiency of the ELM based method.

Recently, researchers mostly used convolutional neural network to extract image features due to the rapid development of deep learning. As for multi-label classification, researchers have proposed some approaches based on the probabilistic graph model or recurrent attention model to deal with the problem. It is worth mentioning that attention mechanisms is also a popular method. Inspired by [9], we propose a novel model based on the graph convolutional network to model the correlations between labels. For example, when the label "Long Hair" occurs, the label "Female" is much more likely to show up than the label "Male". Following this idea, we construct an adjacency matrix between labels to deliver this correlation into classifiers, then combine the image representation to produce multi-label loss. Our code is hosted at https://github.com/2014gaokao/pedestrian-attribute-recognition-with-GCN. The contributions of the paper are as follows:

- This paper creatively applies the novel end-to-end trainable multi-label image recognition framework to pedestrian attribute recognition, which to our knowledge, is the first one tackling pedestrian attribute recognition by graph convolutional network.
- Graph convolutional network normally propagates information between nodes based on correlation matrix. So, we design the correlation matrix for GCN in-depth and propose some improvement methods. Finally, we evaluate our method on pedestrian attribute recognition dataset, and our proposed method consistently achieves superior performance over previous competing approaches.

The result of the paper is organized as follows: Section 2 comprehensively introduces related work in pedestrian attribute recognition. Section 3 describes the overall architecture of our model in detail. Section 4 verified the superiority of our method using experiments. Section 5 concludes the paper and states future research directions.

## 2. Related Work

Pedestrian attribute recognition has attracted a lot of attention from researchers, with many efforts dedicated to extending the deep convolutional network for pedestrian attributes recognition. Li et al. [10] proposed and compared two algorithms, where one only uses deep features for binary classification and the other considers the correlations between human attributes, which proves the importance of the modeling attribute relationship. The PGDM (pose guided deep model) [11] explored the structure knowledge of the pedestrian body for person attributes recognition. They used the pre-trained pose estimation model to extract deep features of part of the regions and fused them with the whole image together to improve final attribute recognition, however, the model could not be end-to-end trainable. Besides, many researchers have put a lot of effort in incorporating attention mechanisms and the sequence model in pedestrian attribute recognition. A HydraPlus-Net was proposed by Liu et al. [12] with the novel multi-directional attention modules to train complex features for fine-grained tasks of pedestrian analysis. The experiment showed that the method achieved significant improvements against the prior methods even if the model was hard to understand. Wang et al. [13] proposed to use the sequence-to-sequence model to assist attribute recognition. They first split the whole image into horizontal strip regions and formed region sequences from top to bottom which could help them better mine region dependency for better recognition performance, but the recognition accuracy was not satisfactory. Zhao et al. [14] considered the recurrent neural network's super capability of learning context correlations and the attention model's super capability of highlighting the region of interest on the feature map to propose two models, i.e., recurrent convolutional, which is used to explore the correlations between different attribute groups with the convolutional-LSTM (long short term memory) model and recurrent attention, which takes the advantage of capturing the interest region, with the results significantly improved. Also, there is never shortage of novel approaches. Dong et al. [15] proposed a curriculum transfer network to handle the issue of less training data. Specifically, they first used the clean source images and their attribute labels online to train the model and then, simultaneously appended harder target images into the model training process to capture harder cross-domain knowledge. Their model was robust for recognizing attributes from unconstrained images taken from-the-wild. Fabbri et al. [16] proposed to use the deep generative model to re-construct super-resolution pedestrian images to deal with the problem of occlusion and low resolution, yet the reconstructed image was not good as expected.

Zhong et al. [17] proposed an image-attribute reciprocal guidance representation method. Due to the relationship between image features and attributes being not fully considered, the author not only investigated image feature and attribute feature together, but also developed a fusion attention mechanism as well as an improvement loss function to address the problem of imbalance attributes. Tan et al. [18] proposed three attention mechanisms including parsing attention, label attention, and spatial attention to highlight regions or pixels against the variations, such as frequent pose variations, blur images, and camera angles. Specifically, parsing attention mainly focuses to extract image features, where label attention pays more attention to attribute features, and spatial attention aims at considering problems from a global perspective, however, they do not fully consider the correlation between attributes. Li et al. [19] proposed to recognize pedestrian attribute by joint visual-semantic reasoning and knowledge distillation while the results remain to be discussed. Han et al. [20] proposed an attention aware pooling method for pedestrian attribute recognition which can also exploit the correlations between attributes. Xiang et al. [21] proposed a meta learning based method for pedestrian attribute recognition to handle the scenario for newly added attributes; semantic similarity and the spatial neighborhood of attributes are not taken into account in this method. In [22], the authors theoretically illustrated that the deeper networks generally take more information into consideration which helps improve classification accuracy. Chen et al. [23] first proposed video-based pedestrian attribute recognition. Their model was divided into two channels: spatial channel extract image features, while the temporal channel took image sequences as input to extract temporal features attached with spatial pooling to integrate the spatial features. Finally, they combed the two channels to achieve attribute classification, but they did not consider the spatial and temporal attention for attribute recognition in videos.

## 3. Approach

In this section, we first introduce some preliminary knowledge of multi-label classification and the graph convolutional network, then we discuss the model in-depth.

### *3.1. Preliminary*

#### 3.1.1. Multi-Label Learning

Traditional supervised learning is prevailing and successful, which is also one of the most studied machine learning paradigms, where each object is just represented by a single feature vector and associated with a single label. But, traditional supervised learning methods have many limitations. In the real world, one object can be represented by many labels and many objects might co-occur in one scenario. The task of multi-label learning is to learn a function which can predict the proper label sets for unseen instances. A naïve way to deal with the multi-label recognition problem is to transform the multi-class into multiple binary-classification problems. However, if a label space contains 20 class labels, then the number of possible label outputs would exceed one million ($2^{20}$). Obviously, we cannot afford the overwhelming exponential-sized output size. So, it is necessary and crucial to capture correlations or dependency among labels which can effectively solve these problems. For example, the probability of an image being annotated with label "Female" would be high if we knew it had labels "Long hair" and "Skirt". For the multi-label classification algorithms, the following three kinds of learning strategies can be concluded as noted in [24]:

- First-order strategy, which directly transform the multi-class into multiple binary-classification problem. This strategy obviously does not take the correlations into consideration and will cause exponential-sized output size;
- Second-order strategy, which only considers the correlations between each label pair; however, label correlations are more complicated than the second-order strategy in real-world applications;
- High-order strategy, which considers all the label relationships by modeling the correlations among labels. Normally, this strategy can achieve better performance but with higher computational complexity.

#### 3.1.2. Graph Convolutional Network

Graph convolutional network is a branch of the graph neural network [25]. It is an emerging field in recent years which originates from the limitations of convolutional neural networks. CNN has developed rapidly these years due to its translation invariance and weight sharing, which started the new era of deep learning [26]. However, the convolutional neural network normally operates on regular Euclidean data-like images and speech, and in other words, convolutional neural networks are not good at operating on non-Euclidean data-like graphs. Therefore, many researchers started to think how to define the convolution on non-Euclidean structures and extract features for machine learning tasks. Advance strategies in graph convolution are often categorized as spectral approaches and spatial approaches. This paper mainly focuses on spectral approaches.

Spectral network was proposed by [27]. The convolution operation is defined in the Fourier domain by computing the eigenvalue decomposition of the graph Laplacian. Given Laplacian:

$$L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^T, \tag{1}$$

where $D$ is the degree matrix, $A$ is the adjacency matrix of the graph, and $\Lambda$ is the diagonal matrix of its eigenvalues. Notice that $L$ is a symmetrical and positive semi-definite matrix which means $L$ can be decomposed and each eigenvalue of $L$ is called the spectrum of $L$. Then, researchers define traditional Fourier transform on graph, the graph Fourier transform of a signal $x \in R^N$ is defined as $U^T x$, where $U$ is the matrix of eigenvectors of the normalized graph Laplacian.

The convolution on graph can be defined as the multiplication of a signal $x \in R^N$ with a filter $g_\theta = diag(\theta)$ parameterized by $\theta \in R^N$:

$$g_\theta * x = U g_\theta U^T x. \tag{2}$$

Then researchers focus on $g_\theta$ and deduce that the computational complexity of the above operation is $O\left(n^3\right)$, which means this operation may result in intense computational complexity. So, researchers hope to come up with a way equipped with fewer parameters and lower complexity. In [28], the author suggests that $g_\theta$ can be approximated by a truncated expansion in term of Chebyshev polynomials $T_k(x)$ up to $K$-th order:

$$g_\theta * x \approx \sum_{k=0}^{K} \theta_k T_k\left(\widetilde{L}\right)x, \tag{3}$$

where $\widetilde{L} = \frac{2}{\lambda_{max}}L - I_N$. $\lambda_{max}$ denotes the largest eigenvalue of $L$. Notice that the approximate operation only requires complexity of $O(K|E|)$ and $K+1$ parameters, where $E$ is the edge numbers of a graph. Next, [29] limits $K = 1$ and approximates $\lambda_{max} \approx 2$ as well as constrains the parameters with $\theta = \theta'_0 = -\theta'_1$ to simplify the operation, then we can obtain the following expression:

$$g_{\theta'} * x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta\left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x, \tag{4}$$

Finally, using renormalization trick, from a macro perspective, the formula can be expressed as follows:

$$H^{l+1} = \sigma\left(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} H^l W^l\right), \tag{5}$$

where $H^l$ denotes the nodes feature of $l$-th layer, $W^l$ denotes the parameters to be learned of $l$-th layer, $\sigma(\cdot)$ denotes a non-linear operation.

## 3.2. Architecture of Our Model

The overall framework of our approach is shown in Figure 1.

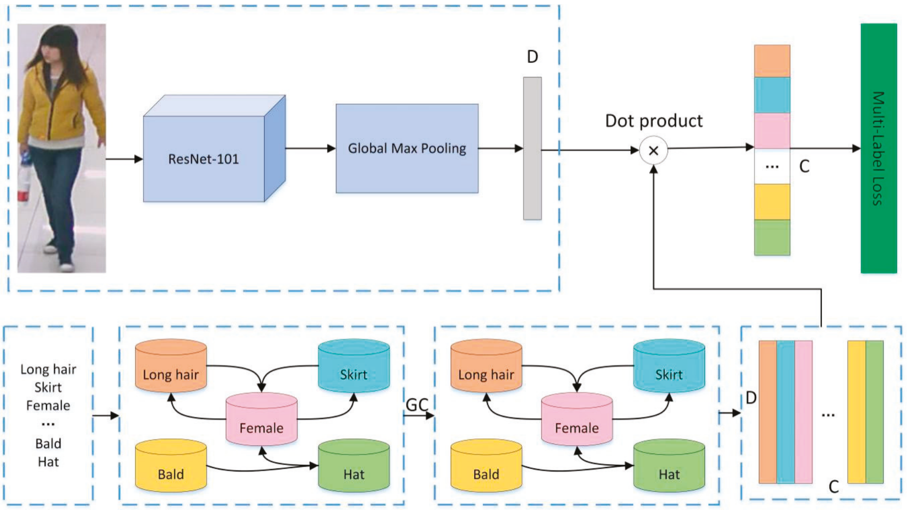


**Figure 1.** Overall framework of our model for pedestrian attribute recognition.

Our model adopts ResNet-101 to extract features of each pedestrian image since ResNet-101 is a common paradigm in image classification, meanwhile, we transform the corresponding attribute labels into word embedding and feed them to our data-driven matrix. The directed line between ellipse pairs represents the dependency of label pairs. Graph convolutional network maps label into D × C-dim classifiers, where D denotes the dimensionality of the parameter to be learned and C denotes the categories of labels. Obviously, our model takes both image and word embedding as input and with multiplication of the corresponding two outputs to produce C-dim scores, and finally, this paper uses traditional multi-label loss to train our network architecture. The detail of image feature extraction and the data-driven matrix is discussed in Sections 3.2.1 and 3.2.2.

### 3.2.1. Image Feature Extraction

Convolutional neural networks have achieved excellent performance in image processing these years; we can use any CNN base models to learn the features of each pedestrian image. In our experiments, this paper uses the deep residual network [30,31] as the base model to push forward our work. Deep residual network originates from the common fault of the deep convolutional neural network, as many researchers used to stack deeper layers to enrich more features and expected to get better recognition performance. However, is learning better networks as easy as stacking more layers? The answer is absolutely no. As layers go deeper at the beginning stage, accuracy will arise steadily, and as the depth continues increasing, the network may tend to get saturated, then the accuracy degrades rapidly and leads to a higher training error. So, researchers have proposed a solution to the awkward situation as shown in Figure 2. The added layers are identity mapping while the other layers are copied from the learned shallower model.

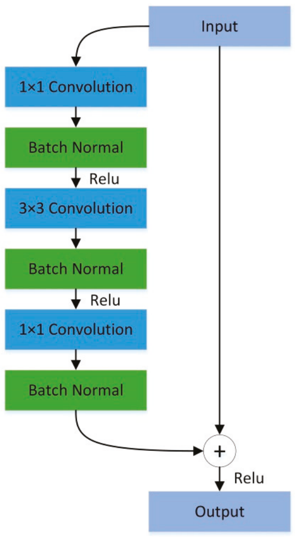


**Figure 2.** Identity mapping.

There are many explanations for why it works. Some researchers suppose that Float 32-bit numbers will no longer represent a gradient change when the network continues backpropagation for a long time once the derivative is less than one at the earlier step, thus shallower networks will not be able to learn things. Instead, deep residual networks take the gradient before derivation into consideration which artificially reduces the probability of gradient dispersion. From my perspective of view, each convolution operation will abandon some information inevitably due to random kernel parameters, inhibition of activation function, and so on. However, residual networks will fetch the information once processed by shortcut to reduce the loss. The authors of the deep residual network have made many model evaluations from 18-layer to 152-layer, and between those are 34-layer, 50-layer, and 101-layer. Our model use ResNet-101 to extract pedestrian image features. We randomly cropped each training image and resized into 224 $\times$ 224 resolution with horizontal flips for data augmentation. Finally, we can obtain 2048 $\times$ 14 $\times$ 14 feature maps from the "conv5_x" layer.

### 3.2.2. Correlation Matrix

Graph convolutional network works by propagating information between nodes based on the correlation matrix. Thus, it is crucial to construct a correlation matrix. Normally, we use adjacent matrix to represent the topology structure of a graph. However, such matrix will not be provided in any benchmark pedestrian attribute recognition datasets. In this paper, to construct the correlation matrix, we first define the matrix $N$:

$$\begin{bmatrix} N_1 & N_2 & \cdots & N_{n-1} & N_n \end{bmatrix}, \tag{6}$$

where $N_i$ denotes the occurrence numbers of each label in the training set, then we count the co-occurrence of label pairs in the training set and get the matrix $M$:

$$\begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1,n-1} & M_{1n} \\ \vdots & & \ddots & & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{n,n-1} & M_{nn} \end{bmatrix}, \tag{7}$$

where $M_{ij}$ denotes the co-occurrence times of $Label_i$ and $Label_j$. Then, we can get the conditional probability matrix by using formula $P_{ij} = M_{ij}/N_i$:

$$\begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1,n-1} & P_{1n} \\ \vdots & & \ddots & & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{n,n-1} & P_{nn} \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1,n-1} & M_{1n} \\ \vdots & & \ddots & & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{n,n-1} & M_{nn} \end{bmatrix} / \begin{bmatrix} N_1 & N_2 & \cdots & N_{n-1} & N_n \end{bmatrix}, \tag{8}$$

where $P_{ij}$ denotes the probability of occurrence of label $L_j$ when label $L_i$ appears. Notice that the formula is not strictly matrix division. The relationship between label pairs is shown as Figure 3, for example, given an image, the probability of occurrence of label "Female" will be very high if label "Skirt" exists, however the probability of occurrence of label "Skirt" might not be high if label "Female" exists, because females wear skirts, but females do not necessarily wear skirts. On the contrary, label "Female" and label "Skirt" both have no relationship with label "Bald", as we seldom see females have no hair.

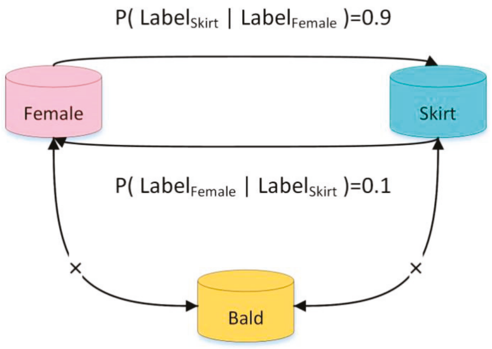

P( Label_Skirt | Label_Female )=0.9
Female
Skirt
P( Label_Female | Label_Skirt )=0.1
×
×
Bald

**Figure 3.** Conditional probability between labels.

## 4. Experiments

In this section, we first introduce the evaluation metrics and dataset as well as implementation details. Then, we report the results on benchmark pedestrian attribute recognition dataset.

### *4.1. Evaluation Metrics*

Following conventional settings, we record five evaluation criteria including accuracy, precision, recall rate, F1 value, and mA. The formula of mA can be calculated as following:

$$\mathrm{mA} = \frac{1}{2\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{L}} \left( \frac{\mathrm{TP_i}}{\mathrm{P_i}} + \frac{\mathrm{TN_i}}{\mathrm{N_i}} \right), \tag{9}$$

where L is the number of attributes. $TP_i$ is the number of correctly predicted positive examples and $TN_i$ is the number of correctly predicted negative examples, $P_i$ is the number of positive examples and $N_i$ is the number of negative examples.

Evaluation metrics on multi-label classification is much more complicated than traditional single-label classification, so there are many evaluation metrics proposed by researchers which can mainly be divided into two groups, i.e., label-based metrics and example-based metrics. Researchers called the above evaluation criterions as label-based evaluation criterions because mA only takes each attribute into consideration and ignores the relationship between attributes. Thus, some researchers suggest using the example-based evaluation criterions like accuracy, precision, recall rate, and F1 value. The formula of those evaluation criteria can be defined as below:

$$\mathrm{Acc_{exam}} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{\left|\mathrm{Y_i} \cap \mathrm{f}(\mathrm{x_i})\right|}{\left|\mathrm{Y_i} \cup \mathrm{f}(\mathrm{x_i})\right|}, \tag{10}$$

$$\mathrm{Prex_{exam}} = \frac{1}{2\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{\left|\mathrm{Y_i} \cap \mathrm{f}(\mathrm{x_i})\right|}{\left|\mathrm{f}(\mathrm{x_i})\right|}, \tag{11}$$

$$\mathrm{Rec_{exam}} = \frac{1}{2\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{\left|\mathrm{Y_i} \cap \mathrm{f}(\mathrm{x_i})\right|}{|\mathrm{Y_i}|}, \tag{12}$$

$$\mathrm{F1} = \frac{2{*}\mathrm{Prec_{exam}}{*}\mathrm{Rec_{exam}}}{\mathrm{Prec_{exam}} + \mathrm{Rec_{exam}}}, \tag{13}$$

where N is the number of examples, $x_i$ denotes for i-th example, f($x_i$) returns the predicted positive labels of $x_i$. $Y_i$ is the ground truth positive labels of the corresponding example.

## 4.2. Dataset

We use RAP-2.0 [32] as our benchmark pedestrian attribute recognition dataset. This dataset contains 84,928 images which were divided into three parts, of which 50,957 for training, 16,986 for validation, and 16,985 for testing. In our experiments, we use both training part and validation part as training dataset. The annotation of each image is very rich as shown in Table 1.

**Table 1.** Annotation attribute names in RAP.

| Class | Attribute |
|---|---|
| Whole body | gender, age, shape, role |
| Head | hair style, hair color, hat, glasses |
| Upper body | clothes style, clothes color |
| Lower body | clothes style, clothes color, shoes style, shoes color |
| Other | bounding box, accessories, actions, occlusion |

## 4.3. Implement Details

We implement ResNet-101 to extract image feature and obtain 2048 × 14 × 14 feature maps from the “conv5_x” layer. Followed by a global max pooling, which size is 14 × 14, can we achieve 2048 × 1 × 1-dim image features. Then we select 60 attributes out of 152 in our benchmark dataset and use one-hot word embedding to transform those attribute labels into 60 × 300-dim word embedding. As the graph convolutional network normally propagates information based on correlation matrix, we count the total occurrence times of each label in the training dataset to construct matrix $N \in R^{1\times60}$ and we also count the occurrence times of label $L_i$ and label $L_j$ in the training set to construct matrix $M \in R^{60\times60}$, then we can get probability matrix $P = M/N \in R^{60\times60}$ as correlation matrix. Thus, graph convolutional network can work by mapping those label representations to classifiers based on the correlation matrix $P$ and we will get an output vector of 2048 × 60 dimensionality.

Obviously, we will get a 60-dim vector by 2048 dot product 2048 × 60. We take the 60-dim vector into traditional multi-label classification loss to ensure our model framework has the ability to be end-to-end trainable.

## 4.4. Experiment Results

As shown in Table 2, our model is a close match against most previous methods before further improvement. To improve final recognition performance, we optimize the correlation matrix, while the details are discussed in Section 4.5. Unsurprisingly, we get a significant improvement according to experiment results. Experiments verified that our improved method goes beyond other competitive counterparts in some of evaluation criteria. Improve operations are discussed in detail in next section.

**Table 2.** Comparison against previous methods.

| Method | RAP | | | | |
|---|---|---|---|---|---|
| **Metric** | **mA** | **Accuracy** | **Precision** | **Recall** | **F1** |
| ACN | 69.66 | 62.61 | 80.12 | 72.26 | 75.98 |
| DeepMar | 73.79 | 62.02 | 74.92 | 76.21 | 75.56 |
| HydraPlus-Net | 76.12 | 65.39 | 77.33 | 78.79 | 78.05 |
| JRL | 77.81 | - | 78.11 | 78.98 | 78.58 |
| VeSPA | 77.70 | 67.35 | 79.51 | 79.67 | 79.59 |
| Ours | 74.04 | 65.78 | 80.12 | 76.88 | 78.47 |
| Ours-improved | 75.97 | **68.99** | **81.48** | **79.97** | **80.72** |

## 4.5. Ablation Study

In order to further improve final recognition performance, we propose to binarize the correlation matrix $P$, because $P_{ij}$ may be a noisy edge if $P_{ij}$ has a very small value. Specifically, we use the threshold $t$ to filter noisy edges:

$$A_{ij} = \begin{cases} 0, & if\ P_{ij} < t \\ 1, & if\ P_{ij} \geq t \end{cases} . \tag{14}$$

As shown in Figure 4, it will achieve a better recognition performance than the initial model when the threshold is about 0.6 and obviously it gets top results when the threshold is about 0.9. But it may cause sparse problems, for example, if t = 0.9, then most of edges will be filtered, so we propose to assign proportion $p$, which means the weight between the node itself and its neighbor nodes are:

$$A_{ij} = \frac{p}{\sum_{j=1}^{C} A_{ij} + \varepsilon} A_{ij}, \tag{15}$$

where ε is a very small number, notice that if the proportion is too small, it will ignore the information from the neighbor nodes; if proportion is too large, it will ignore the feature of the node itself. As shown in Figure 5, our model will get better performance if proportion is close to 0.5, but the proportion strategy does not work for improving overall performance.

Furthermore, we take the GCN layers into consideration to see if the deeper layer can get higher accuracy. We stack more layers to see how the layer numbers affect the model performance. As shown in Figure 6, our model achieves best performance when layer equals to 3. As shown, when the number of graph convolution layers increases over 4, recognition performance drops quickly. The possible reason for the performance drop may be that when using more GCN layers, the propagation between nodes will be accumulated, which can result in over-smoothing.

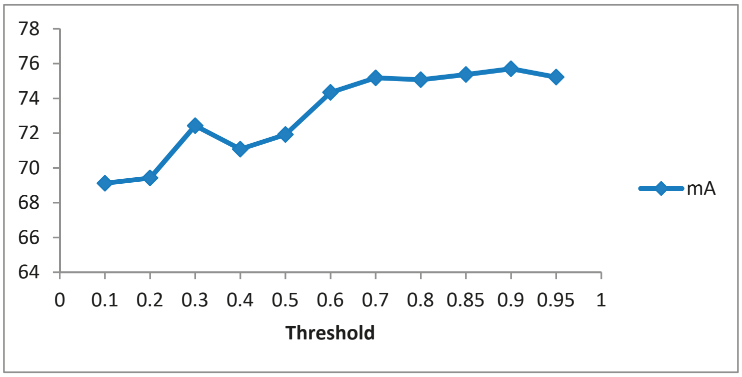


**Figure 4.** Accuracy comparison of different thresholds.

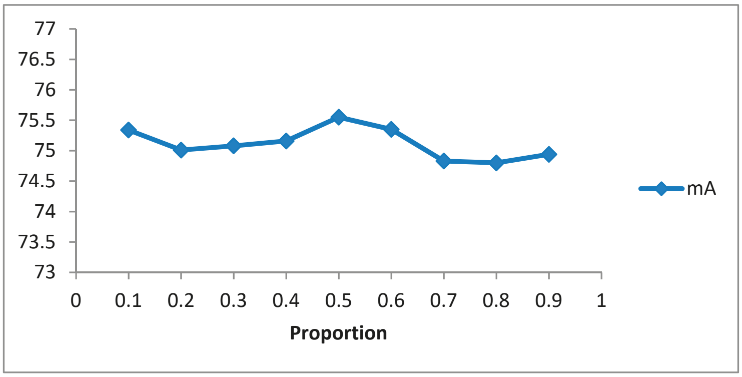


**Figure 5.** Accuracy comparisons of different proportions.

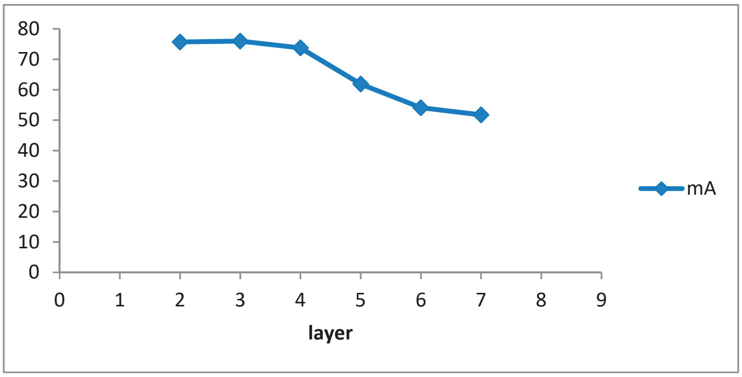


**Figure 6.** Accuracy comparisons of different layers.

### *4.6. Image Retrieval*

For a more intuitive display, we wrote a program to retrieve the pedestrian image in test dataset with some query words through our model. As shown in Figure 7, we can clearly observe that our image representation results are pretty accurate.

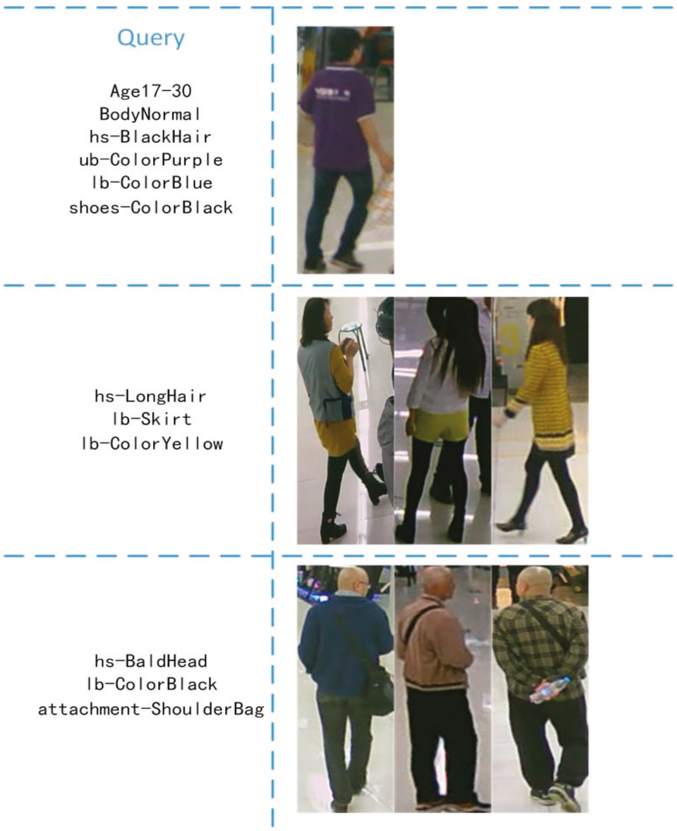


**Figure 7.** Query results.

## 5. Conclusions

Our model overturns conventional approaches, instead we apply a novel framework based on the graph convolutional network for pedestrian attribute recognition and obtain satisfactory results. Given a pedestrian image and corresponding labels, we choose ResNet-101 to extract image features, and transform attribute labels to word embedding, then we construct a correlation matrix according to the occurrence of labels in the training dataset and optimize the matrix for better performance. Graph convolutional network helps us map the information between nodes into classifiers, incorporating the image features extracted by convolutional neural network to ensure our model has the ability to be end-to-end training, which is graceful for network architecture and amazing for final recognition performance. Experiments have validated the superiority of our model.

**Author Contributions:** Conceptualization, C.Z.; software, C.Z.; validation, X.S., H.Y.; formal analysis, X.S.; writing—original draft preparation, X.S.; writing—review and editing, C.Z.

**Funding:** This research received no external funding.

**Acknowledgments:** The authors thank all the anonymous reviewers for their insightful comments and useful suggestions.

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

## References

1. Porikli, F.; Bremond, F.; Dockstader, S.L.; Ferryman, J.; Hoogs, A.; Lovell, B.C.; Pankanti, S.; Rinner, B.; Tu, P.; Venetianer, P.L. Video surveillance: Past, present, and now the future. *IEEE Signal Process. Mag.* **2013**, *30*, 190–198. [CrossRef]
2. Li, Q.; Zhao, X.; He, R.; Huang, K. Visual-semantic graph reasoning for pedestrian attribute recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
3. Wang, Y.; Gan, W.; Wu, W.; Yan, J. Dynamic Curriculum Learning for Imbalanced Data Classification. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
4. Wang, X. Pedestrian Attribute Recognition: A Survey. *arXiv* **2019**, arXiv:1901.07474.
5. Huang, G.; Huang, G.-B.; Song, S.; You, K. Trends in extreme learning machines: A review. *Neural Netw.* **2015**, *61*, 32–48. [CrossRef] [PubMed]
6. Salerno, V.M.; Rabbeni, G. An Extreme Learning Machine Approach to Effective Energy Disaggregation. *Electronics* **2018**, *7*, 235. [CrossRef]
7. Hussain, T.; Siniscalchi, S.M.; Lee, C.-C.; Wang, S.-S.; Tsao, Y.; Liao, W.-H. Experimental study on extreme learning machine applications for speech enhancement. *IEEE Access* **2017**, *5*, 25542–25554. [CrossRef]
8. Yang, K.; Du, E.Y.; Delp, E.J.; Jiang, P.; Jiang, F.; Chen, Y.; Sherony, R.; Takahashi, H. An Extreme Learning Machine-based pedestrian detection method. In Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, Australia, 23–26 June 2013.
9. Chen, Z.; Wei, X. Multi-Label Image Recognition with Graph Convolutional Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR, Long Beach, CA, USA, 16–20 June 2019.
10. Li, D.; Chen, X.; Huang, K. Multi-attribute learning for pedestrian attribute recognition in surveillance scenarios. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 111–115.
11. Li, D.; Chen, X.; Zhang, Z.; Huang, K. Pose guided deep model for pedestrian attribute recognition in surveillance scenarios. In Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, USA, 23–27 July 2018; pp. 1–6.
12. Liu, X.; Zhao, H.; Tian, M.; Sheng, L.; Shao, J.; Yi, S.; Yan, J.; Wang, X. Hydraplus-net: Attentive deep features for pedestrian analysis. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 350–359.
13. Wang, J.; Zhu, X.; Gong, S.; Li, W. Attribute recognition by joint recurrent learning of context and correlation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 531–540.
14. Zhao, X.; Sang, L.; Ding, G.; Han, J.; Di, N.; Yan, C. Recurrent attention model for pedestrian attribute recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
15. Dong, Q.; Gong, S.; Zhu, X. Multi-task curriculum transfer deep learning of clothing attributes. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017.
16. Fabbri, M.; Calderara, S.; Cucchiara, R. Generative adversarial models for people attribute recognition in surveillance. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017.
17. Zhong, J.; He, E.; Wang, H.; Yang, A. Image-attribute reciprocally guided attention network for pedestrian attribute recognition. *Pattern Recognit. Lett.* **2019**, *120*, 89–95.
18. Tan, Z.; Yang, Y.; Wan, J.; Chen, Y.; Guo, G.; Li, S. Attention based Pedestrian Attribute Analysis. *IEEE Trans. Image Process.* **2019**, *28*, 6126–6140. [CrossRef] [PubMed]
19. Li, Q.; Zhao, X.; He, R.; Huang, K. Pedestrian Attribute Recognition by Joint Visual-semantic Reasoning and Knowledge Distillation. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
20. Han, K.; Wang, Y.; Shu, H.; Liu, C.; Xu, C.J.; Xu, C. Attribute Aware Pooling for Pedestrian Attribute Recognition. *arXiv* **2019**, arXiv:1907.11837.

21. Xiang, L.; Jin, X.; Ding, G.; Han, J.; Li, L. Incremental Few-Shot Learning for Pedestrian Attribute Recognition. *arXiv* **2019**, arXiv:1906.00330.
22. Bekele, E.; Lawso, W. The Deeper, the Better: Analysis of Person Attributes Recognition. *arXiv* **2019**, arXiv:1901.03756.
23. Chen, Z.; Li, A.; Wang, Y. Video-Based Pedestrian Attribute Recognition. *arXiv* **2019**, arXiv:1901.05742.
24. Zhang, M.-L.; Zhou, Z.-H. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1819–1837. [CrossRef]
25. Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph Neural Networks: A Review of Methods and Applications. *arXiv* **2018**, arXiv:1812.08434.
26. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef] [PubMed]
27. Bruna, J.; Zaremba, W.; Szlam, A.; Lecun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
28. Hammond, D.K.; Vandergheynst, P.; Gribonval, R. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* **2011**, *30*, 129–150. [CrossRef]
29. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2015.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2016.
32. Li, D.; Zhang, Z.; Chen, X.; Huang, K. A richly annotated pedestrian dataset for person retrieval in real surveillance scenarios. *IEEE Trans. Image Process.* **2019**, *28*, 1575–1590. [CrossRef] [PubMed]

*future internet*

*Article*

# Partitioning Convolutional Neural Networks to Maximize the Inference Rate on Constrained IoT Devices

**Fabíola Martins Campos de Oliveira * and Edson Borin ***

Institute of Computing, University of Campinas, Campinas 13083-852, SP, Brazil
* Correspondence: fabiola.oliveira@ic.unicamp.br (F.M.C.d.O.); borin@unicamp.br (E.B.)

Received: 3 September 2019; Accepted: 26 September 2019; Published: 29 September 2019

**Abstract:** Billions of devices will compose the IoT system in the next few years, generating a huge amount of data. We can use fog computing to process these data, considering that there is the possibility of overloading the network towards the cloud. In this context, deep learning can treat these data, but the memory requirements of deep neural networks may prevent them from executing on a single resource-constrained device. Furthermore, their computational requirements may yield an unfeasible execution time. In this work, we propose Deep Neural Networks Partitioning for Constrained IoT Devices, a new algorithm to partition neural networks for efficient distributed execution. Our algorithm can optimize the neural network inference rate or the number of communications among devices. Additionally, our algorithm accounts appropriately for the shared parameters and biases of Convolutional Neural Network. We investigate the inference rate maximization for the LeNet model in constrained setups. We show that the partitionings offered by popular machine learning frameworks such as TensorFlow or by the general-purpose framework METIS may produce invalid partitionings for very constrained setups. The results show that our algorithm can partition LeNet for all the proposed setups, yielding up to 38% more inferences per second than METIS.

**Keywords:** Internet of Things; convolutional neural networks; graph partitioning; distributed systems; resource-efficient inference

## 1. Introduction

In the next few years, a burst in the number of Internet-of-Things (IoT) devices is expected [1–3]. IoT devices present many sensors and can generate a large amount of data per second, which will prevent these data from being sent to the cloud for processing due to the high and variable latency and limited bandwidth of current networks [1,4]. Thus, an approach to process the large amount of data generated by the IoT and to efficiently use the IoT limited resources is fog computing, which allows the applications or part of them to be executed closer to the devices or on the devices themselves [5].

To achieve the billions of devices estimated for the IoT system, many of them will have to be constrained, for instance, in size and cost. A constrained device presents limited hardware in comparison to the current devices connected to the Internet. Recently, a classification of constrained devices has been proposed, showing the increasing importance of them in the IoT [6]. These devices are constrained due to their embedded nature and/or size, cost, weight, power, and energy. Considering that these constraints impact on the amount of memory, computational power, communication performance, and battery life, these resources must be properly employed to satisfy applications requirements. The proposed classification not only differentiates more powerful IoT devices such as smartphones and single-board computers such as Raspberry Pi from constrained devices but also delimits the IoT scope, which does not include servers either desktop or notebook computers.

To obtain valuable information from the vast amount of data generated by the IoT, deep learning can be used since it can extract automatic features from the data and strongly benefits from large amounts of data [7]. Nevertheless, deep learning techniques often present a high computational cost, which brings more challenges in using resource-limited devices even if we only consider executing the inference phase of these methods. These constrained devices may impact an application that has as requirements real-time responses or a high inference rate, for instance.

The size and computational requirements of current Deep Neural Networks (DNNs) may not fit constrained IoT devices. Two approaches are commonly adopted to enable the execution of DNNs on this type of device. The first approach prunes the neural network model so that it requires fewer resources. The second approach partitions the neural network and executes in a distributed way on multiple devices. In some works that employ the first approach, pruning a neural network results in accuracy loss [8–10]. On the other hand, several works can apply the first approach to reduce DNN requirements and enable its execution on limited devices without any accuracy loss [11–13]. However, it is important to notice that, even after pruning a DNN, its size and computational requirements may still prevent the DNN from being executed on a single constrained device. Therefore, our focus is on the second approach. In this scenario, the challenge of how to distribute the neural network aiming to satisfy one or more requirement arises.

Some Machine Learning (ML) and IoT frameworks that offer the infrastructure to distribute the neural network execution to multiple devices already exist such as TensorFlow, Distributed Artificial Neural Networks for the Internet of Things (DIANNE), and DeepX [14–16]. However, they require the user to manually partition the neural network and they limit the partitioning into a per-layer approach. The per-layer partitioning may prevent neural networks from being executed on devices with more severe constraint conditions, for instance, some devices from the STM32 32-bit microcontroller family [17]. This may happen because there may be a single DNN layer whose memory requirements do not fit the available memory of these constrained devices. On the other hand, other general-purpose, automatic partitioning tools such as SCOTCH [18] and METIS [19] do not take into account the characteristics of neural networks and constrained devices. For this reason, they provide a suboptimal result or, in some cases, they are not able to provide any valid partitioning.

Recently, we proposed Kernighan-and-Lin-based Partitioning [20], an algorithm to automatically partition neural networks into constrained IoT devices, which aimed to reduce the number of communications among partitions. Communication reduction is important so that the network is not overloaded, a situation that can be aggravated in a wireless connection shared with several devices. Even though reducing communication may help any system, in several contexts, one of the main objectives is to optimize (increase) the inference rate, especially on applications that need to process a data stream [5,21–23].

In this work, we extend this preliminary work and propose Deep Neural Networks Partitioning for Constrained IoT Devices (DN$^2$PCIoT), an algorithm to automatically partition DNNs into constrained IoT devices, including inference rate maximization or communication reduction as objective functions. Additionally, for both objective functions, this new algorithm accounts more precisely for the amount of memory required by the shared parameters and biases of Convolutional Neural Networks (CNNs) in each partition. This feature allows our algorithm to provide valid partitionings even when more constrained setups are employed in the applications.

We are concerned with scenarios in which data are produced within constrained devices and only constrained devices such as the ones containing microphones and cameras are available to process these data. Although constrained devices equipped with cameras might not be constrained in some of their resources, we have to consider that only part of these resources is available for extra processing. After all, the devices have to execute their primary task in the first place.

Several IoT resources can be considered when designing an IoT solution to improve quality of service. The main IoT issues include the challenges in the network infrastructure and the large amount of data generated by the IoT devices, but other requirements such as security, dependability, and energy consumption are equally important [24]. Additionally, minimizing communication is important to reduce interference in the wireless medium and to reduce the power consumed by radio operations [25]. These issues and requirements usually demand a trade-off among the amount of memory, computational power, communication performance, and battery life of the IoT devices. For instance, by raising the levels of security and dependability, offloading processing to the cloud, and/or processing data on the IoT devices, energy consumption is raised as well, impacting on the device battery life. In this work, we are concerned with the requirement that many DNN applications presents: the DNN inference rate maximization. Our objective is to treat the large amount of data generated by the IoT devices by executing DNNs on the devices themselves. We also address some of the challenges in the network infrastructure by reducing communication between IoT devices.

We use the inference rate maximization objective function to partition the LeNet CNN model using several approaches such as per-layer partitionings provided by popular ML frameworks, partitionings provided by METIS, and by our algorithm DN$^2$PCIoT. We show that DN$^2$PCIoT starting from random partitionings or DN$^2$PCIoT starting from partitionings generated by the other approaches results in partitionings that achieve up to 38% more inferences per second than METIS. Additionally, we also show that DN$^2$PCIoT can produce valid partitionings even when the other approaches cannot. The main contributions of this article are summarized as follows:

- the DN$^2$PCIoT algorithm that optimizes partitionings aiming for inference rate maximization or communication reduction while properly accounting for the memory required by the CNNs' shared parameters and biases;
- a case study whose results show that the DN$^2$PCIoT algorithm is capable of producing partitionings that achieve higher inference rates and that it is also capable of producing valid partitionings for very constrained IoT setups;
- a case study of popular ML tools such as TensorFlow, DIANNE, and DeepX, which may not be able to execute DNN models on very constrained devices due to their per-layer partitioning approach;
- a study of the METIS tool, which indicates that it is not an appropriate tool to partition DNNs for constrained IoT setups because it may not provide valid partitionings under these conditions;
- an analysis of the DNN model granularity results to show that our DNN with more grouping minimally affects the partitioning result;
- an analysis of how profitable it is to distribute the inference rate execution among multiple constrained devices; and
- a greedy algorithm to reduce the number of communications based on the available amount of memory of the devices.

This paper is organized as follows. Section 2 provides the background in CNNs and neural networks represented as a dataflow graph; it also presents the related work in ML and IoT tools and in general-purpose, automatic partitioning algorithms. Section 3 presents the DN$^2$PCIoT algorithm. Section 4 explains how LeNet was modeled, the adopted approaches, and the experiment setups. Section 5 presents and discusses the results. Finally, Section 6 provides the conclusions.

## 2. Background and Related Work

In this section, the background in CNNs and important concepts in modeling neural networks as a dataflow graph are discussed, as well as the related work in specific ML and IoT tools and general-purpose partitioning algorithms.

### *2.1. Convolutional Neural Network*

CNNs are composed of convolution layers, pooling layers, and fully connected layers [26]. The pooling layers transform the high-resolution input data into a coarser resolution and also make the input invariant to translations. At the neural network end, a fully connected layer indeed classifies the input. CNNs arrange the neurons of each layer in three dimensions: height, width, and depth.

The LeNet model that we used in this work was the first successful CNN, which was first applied to recognize handwritten digits in images [27]. However, it can be applied to other kinds of recognition as well [28]. In convolution layers, there is a set of shared parameters and biases for each layer, which is shared among all the neurons of that layer. For pooling layers, in this version of LeNet, there is a set of biases and trainable coefficients for each layer, which is also shared among all the neurons of that layer. In fully connected layers, in this version of LeNet, each neuron has its own parameter set and bias.

### *2.2. Dataflow Graphs and Neural Network Models*

Some important concepts need to be defined before proceeding with the related work in ML, IoT, and partitioning tools. Neural networks can be modeled as a dataflow graph. Dataflow graphs are composed of a directed acyclic graph that models the computation of a program through its data flow [29]. In a dataflow graph, vertices represent computations and may send/receive data to/from other vertices in the graph. In our approach, a vertex represents one or more neural network neurons and may also require an amount of memory to store the intermediate (layer) results and the neural network parameters required by the respective neurons it represents. Dataflow graph edges may contain weights to represent different amounts of data that are sent to other vertices.

Figure 1a shows a simple fully connected neural network represented as a dataflow graph. In this graph, each dataflow vertex represents one neural network neuron. The first layer is the input layer with two vertices; each vertex requires 4 bytes (B) to store the neuron input value, if we use data represented by 4 B. The second layer is the hidden fully connected layer; each vertex requires 12 B, being 4 B to store the neuron intermediate result and the other 8 B to store the neuron parameters, which are the edge weights that are multiplied by each input value. It is worth noting that, in this example, no bias is used, so the bias weight is not needed. Furthermore, in the case of CNNs, there is only one set of parameters per layer in the case of convolution layers and not parameters per neurons as in this example. Each vertex in this layer performs 4 floating-point operations (FLOP) per inference, which correspond to the multiplication of the input values by the parameters, to the sum of both multiplied values, and the application of a function to this result. The last layer is a fully connected output layer that contains one vertex; this vertex requires 16 B, being 4 B to store the final result and the other 12 B to store the neuron parameters. It performs 6 FLOP, which correspond to the three multiplications of the parameters by the layer input values, to the two sums of the multiplied values, and the application of a function to this result.

Figure 1b shows the same dataflow graph partitioned for distributed execution on two fictional devices: device A, which can perform 18 FLOP/second (FLOP/s) and provide 20 B of memory and device B, which can perform 18 FLOP/s and provide 52 B of memory. Additionally, the communication link between these devices can transfer 4 B per second. The amount of transferred data per inference in this partitioning is 8 B because, although six edges are crossing the partitions, they represent the data transfer of only 8 B.

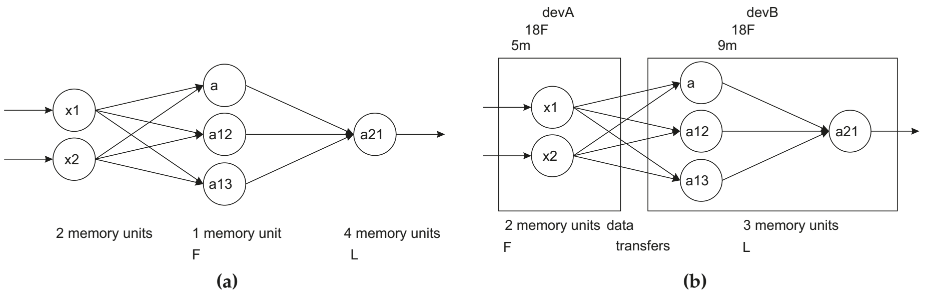


**Figure 1.** Example of: (**a**) how a fully connected neural network may be represented as a dataflow graph; and (**b**) how it can be partitioned for execution on two devices.

We define the cost of a partitioning as the calculation of the objective (or cost) function for that partitioning. If we want to optimize the neural network for the inference rate, then this cost is the inference rate calculation for the partitioning that we have at hand. Since all devices and communication links can work in parallel, the inference rate of a partitioned neural network can be calculated as the minimum value between the inference rate of the devices and the inference rate of the communication links between each pair of devices, according to

$$\text{inference rate} = \min(\text{inference rate}_{devices}, \text{inference rate}_{links}). \tag{1}$$

The inference rate of the devices is calculated as the minimum value between each device computational power divided by the total computational requirement of the vertices that compose the partition assigned to that device:

$$\text{inference rate}_{devices} = \min\left[\left(\frac{\text{computational power}}{\text{computational load}}\right)_d\right], \forall d \in 1, ..., p, \tag{2}$$

in which $p$ is the number of devices in the system. The inference rate of the communication links between each pair of devices is calculated as the minimum value between the transfer performance of each link divided by the total communication requirement of the two partitions involved in this link:

$$\text{inference rate}_{links} = \min\left[\left(\frac{\text{link performance}}{\text{communication load}}\right)_{dq}\right], \forall d, q \in 1, ..., p, \tag{3}$$

in which $dq$ represents the communication link between devices $d$ and $q$.

Thus, taking into account the previous equations, in the partitioning of Figure 1b, device A can perform $18/0 = \infty$ inferences/s, which means device A does not limit the inference rate. The communication link between device A and device B can perform $4/8 = 0.5$ inference/s. Device B can perform $18/18 = 1$ inference/s. Therefore, the inference rate of this partitioning is 0.5 inference/s, which is the minimum value among the inference rate of the devices and the communication links. It is worth noting that this partitioning is valid because both partitions respect the memory limit of the devices.

## 2.3. Problem Definition

In this subsection, we formally define the partitioning problem as a partitioning objective-function optimization problem subject to constraints. First, we define a function that returns 1 if an element $n$ is assigned to partition $p$ and 0 otherwise:

$$partition(p,n) = \begin{cases} 1, & \text{if } n \text{ is assigned to } p; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The partitioning problem can be defined as a partitioning objective-function optimization problem subject to memory constraints:

$$\begin{array}{ll} \text{optimize} & cost \\ \text{subject to} & \sum_{n=1}^{N} m_n \times partition(p,n) + \sum_{j=1}^{L} m_{sbp_j} \times partition(p,j) \leq m_p, \forall p \in [1...P], \end{array} \tag{5}$$

in which *cost* is the objective function (detailed below), $N$ is the number of neurons in the DNN, $m_n$ is the memory required by element $n$, $L$ is the number of layers of the DNN, and $sbp$ is the shared parameters and biases.

If we want to reduce communication, we can define a function that returns 1 if two elements are assigned to different partitions and 0 otherwise:

$$\text{diff}(i,j) = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are assigned to different partitions;} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Then, we can define the communication cost as

$$\text{communication cost} = \sum_{i=1}^{N} \sum_{j=1}^{adj(i)} \text{edge weight}_{ij} \times \text{diff}(i,j), \tag{7}$$

in which $adj(i)$ are the adjacent neurons of neuron $i$ and edge weight$_{ij}$ is the edge weight between neurons $i$ and $j$.

If we want to maximize the inference rate, then Equation (1) represents the cost function and, to formally define the optimization problem, we can rewrite the computational load of device $d$ of Equation (2) as

$$\text{computational load}_d = \sum_{i=1}^{N} \text{computational load}_i \times partition(d,i), \tag{8}$$

and the communication load between devices $d$ and $q$ of Equation (3) as

$$\text{communication load}_{dq} = \sum_{i=1}^{N} \sum_{j=1}^{adj(i)} \text{edge weight}_{ij} \times \text{diff}(i,j) \times partition(d,i) \times partition(q,j). \tag{9}$$

### *2.4. Machine Learning and IoT Tools*

When dealing with the problem of deploying deep learning models on IoT devices, two approaches are commonly used: either the neural network is reduced so that it fits constrained devices (the neural network can use fewer neurons and/or fewer parameters) or the neural network execution is distributed among more than one device, which is an approach that may present performance issues.

One approach to reducing the neural network size to enable its execution on IoT devices is the Big-Little approach [8]. In this approach, a small, critical neural network is obtained from the original DNN to classify the most important classes that should be identified in real time such as the occurrence of fire in a room. For other noncritical classes, data are sent to the cloud for inference in the complete neural network. This approach depends on the cloud for the complete inference and presents some accuracy loss.

Some accuracy loss also happens in the work proposed by Leroux et al. [10], which build several neural networks with an increasing number of parameters. Their approach is called Multi-fidelity DNNs. The neurons of these neural networks are designed to match different IoT devices according to their computational resources. This design aims to satisfy the heterogeneity of IoT systems. However, there is some accuracy loss for each version of the original neural network that they used. This loss may not be acceptable under some circumstances.

DeepIoT proposes a unified approach to compress DNNs that works for CNNs, fully connected neural networks, and recurrent neural networks [13]. The compression makes smaller dense matrices by extracting redundant neurons from the DNN. It can greatly reduce the DNN size, which also greatly reduces the execution time and energy consumption without loss of accuracy. However, as discussed in the Introduction, even after pruning a DNN, its requirements may still prevent it from being executed on a single constrained device. Thus, this approach may not be sufficient and we focus on distributing the execution of DNNs to multiple constrained devices.

Regarding the distributed execution of neural networks, TensorFlow is the Google ML framework that distributes both the training and the inference of neural networks among heterogeneous devices, ranging from mobile devices to large servers [14]. The partitioning must be defined by the user, which is limited to a per-layer fashion to enable the use of TensorFlow's implemented functions. The per-layer partitioning not only produces suboptimal results [20] but also cannot be deployed on very constrained devices. Additionally, TensorFlow aims to speed up the training of neural networks and does not consider the challenges of constrained IoT systems, for instance, memory, communication, computation, and energy requirements.

Distributed Artificial Neural Networks for the Internet of Things (DIANNE) is an IoT-specific framework that models, trains, and evaluates neural networks distributed among multiple devices [15]. The tool is optimized for streaming inference, but here again, the user must manually partition the model into layers, which may limit the performance and may not work for very constrained scenarios.

When it is not possible to run an application on a single IoT device, another approach is to offload some parts of the code onto the cloud. DeepX is a hybrid approach that not only reduces the neural network size but also offloads the execution of some neural network layers onto the cloud, dynamically deciding between its local CPU, GPU, or the cloud itself [16]. Besides the fact that the DeepX runtime may be computationally too heavy to run on constrained devices that are more constrained than smartphones, the model must be partitioned into layers again. Additionally, DeepX may not be able to distribute the neural network to other local devices.

The code offloading approach was also used by Benedetto et al. [30] in a framework that decides if some general computation should be executed locally or should be offloaded onto the cloud. Although this approach is interesting, as well as the fact that constrained IoT devices may prevent their runtime program to execute on such a small device, in this work, we are considering a scenario in which it is not possible to send data to the cloud all the time and we have only constrained devices that can perform the inference of DNNs.

Li, Ota, and Dong [31] proposed the opposite situation: a tool to offload deep learning on cloud computing onto edge computing, i.e., deep learning processing that would be first executed on the cloud can also be offloaded onto IoT gateways and other edge devices. This offload aims to improve learning performance while reducing network traffic, but it also employs a per-layer approach.

Finally, Zhao, Barijough, and Gerstlauer [32] proposed DeepThings, a framework for the inference distribution with a partitioning along the neural network data flow to resource-constrained IoT edge devices. However, they used a small number of devices and a high amount of memory, avoiding the use of more constrained devices such as the ones used in this work.

We summarize all the ML and IoT tools discussed in this subsection in Table 1 with their main characteristics.

**Table 1.** Summary of ML and IoT tools discussed in the related work.

| Approach | Reduce DNN to Fit Constrained Memory? | Loss of Accuracy? | Offload to the Cloud? | Partitioning Type | Type |
|---|---|---|---|---|---|
| Big-Little [8] | Yes | Yes | Yes | per layers | ML IoT |
| DIANNE [15] | No | No | No | per layers | ML IoT |
| DeepX [16] | Yes | Yes | Yes | per layers | ML IoT |
| TensorFlow [14] | No | No | No | per layers ** | ML |
| DeepIoT [13] | Yes | No | No | N/A * | ML IoT |
| Li, Ota, and Dong [31] | No | No | Yes | per layers | ML IoT |
| DeepThings [32] | No | No | No | along the neural network layers | ML IoT |
| Multifidelity [10] | Yes | Yes | No | N/A | ML IoT |
| Benedetto et al. [30] | No | No | Yes | per neurons | IoT |

* Not applicable. ** To use implemented functions.

## 2.5. Partitioning Algorithms

As explained above, the computation distribution may affect inference performance. One solution to avoid these issues is to use automatic, general-purpose partitioning algorithms to define a profitable partitioning for the DNN inference. One of the tools to do that is SCOTCH, which performs graph partitioning and static mapping [18]. The goal of this tool is to balance the computational load while reducing communication costs. However, as SCOTCH was not designed for constrained devices, there is no memory constraint treatment and it may produce invalid partitionings. Additionally, this tool cannot factor redundant edges out, which are edges that represent the same data transfer to the same partition, a situation that often happens in partitioned neural networks.

Kernighan and Lin originally proposed an algorithm [33] to partition graphs that has a large application in distributed systems [34–36]. First, their heuristic randomly partitions a graph that may represent the computation of some application among the partitions. Then, the algorithm calculates the communication cost for this random initial partitioning and tries to improve it by swapping vertices from different partitions and calculating the gain or loss in performing this swap. The best swap operation in each iteration is chosen and its respective vertices are locked for the next iterations and cannot be selected anymore until every pair is selected. When every pair is selected, the whole process may be repeated while improvements are made so that it is possible to achieve a near-optimal partitioning, according to the authors. This algorithm also accounts for partition balance in the hope of achieving an adequate performance while reducing communication.

Another tool is METIS, an open-source library and software from the University of Minnesota that partitions large graphs and meshes and also computes orderings of sparse matrices [19]. This tool employs an algorithm that partitions graphs in a multilevel way, i.e., first, the algorithm gradually groups the graph vertices based on their adjacency until the graph presents only hundreds of vertices. Then, the algorithm applies some partitioning algorithm such as Kernighan and Lin [33] to the small graph and, finally, returns to the original graph also in a multilevel way, performing refinements with the vertices of the edges of the partitions during this return. METIS also reduces communication while balances all the other constraints, which may be memory and computational load, for instance. However, METIS does not present an appropriate treatment of memory constraints either and, thus, may produce invalid partitionings. Additionally, METIS cannot eliminate redundant edges either.

A multilevel Kernighan and Lin approach was developed aiming to achieve the near-optimal solutions of Kernighan and Lin and the fast execution time of METIS to partition software components in mobile cloud computing [37]. This solution takes into account the system heterogeneity and local devices but does not consider memory constraints or redundant edges. Furthermore, the aim is to minimize bandwidth (by reducing weighted communication), which may not yield the best result for other objective functions such as inference rate. This solution is fast but sacrifices the bandwidth result.

All the general-purpose approaches discussed so far in this subsection are edge-cut partitionings, i.e., the algorithms partition the graph vertices into disjoint subsets [38]. Another strategy to general-purpose graph partitioning is vertex-cut partitioning, which partitions the graph edges into disjoint subsets, while the vertices may be replicated among the partitions. Rahimian et al. [39] proposed JA-BE-JA-VC, an algorithm that performs vertex-cut partitioning. Their approach attempts to balance the partitioning aiming to satisfy memory constraints. The main disadvantage of this approach is that it needs vertex replicas, that is, computation replicas, and synchronization, which may involve more communication. When we consider constrained IoT devices and their computational performance, the computation replicas may decrease the inference rate of neural networks to a value that does not comply with the application requirements. As this algorithm is for general purpose, it also does not eliminate redundant edges and does not account for the shared parameters and biases of CNNs adequately.

The tools presented in this section may be useful for distributed execution of neural networks, although the ML frameworks do not present an automatic, flexible partitioning and the general-purpose partitioning algorithms do not treat memory restrictions, redundant edges, shared parameters, and biases properly. We summarize the partitioning algorithms discussed in this subsection in Table 2 with their main characteristics. The next section presents the proposed DN$^2$PCIoT and discusses how we deal with these issues.

**Table 2.** Summary of partitioning algorithms discussed in the related work.

| Approach | Memory Constraints? | Partition Balance? | Eliminate Redundant Edges? | Objective-Function | Adequate Account of Shared Parameters? |
|---|---|---|---|---|---|
| SCOTCH [18] | No | With some load unbalancing | No | Reduce communication | No |
| KL [33] | Yes | With some unbalancing | No | Reduce communication | No |
| METIS [19] | No | With some unbalancing in the constraints | No | Reduce communication | No |
| Multilevel KL [37] | Yes | No | No | Reduce communication | No |
| JA-BE-JA-VC [39] | No | Yes | No | Balance partitions | No |
| Our approach (DN$^2$PCIoT) | Yes | No | Yes | Maximize inference rate or reduce communication | Yes |

## 3. Proposed Deep Neural Networks Partitioning for Constrained IoT Devices (DN$^2$PCIoT)

The DN$^2$PCIoT algorithm is inspired by the Kernighan and Lin's approach, which attempts to find a better solution than its initial partitioning by swapping vertices from different partitions. The Kernighan and Lin's algorithm avoids some local minimum solutions by allowing swaps that produce a partitioning that is worse than the previous one. This situation can happen if such a swap is the best operation at some point in the algorithm.

DN$^2$PCIoT accepts a dataflow graph as the input for the neural network, in which the vertices represent the neural network neurons (input data, operations, or output data), and the edges represent data transfers between the vertices. This same approach is used in SCOTCH and METIS. DN$^2$PCIoT also receives a target graph, which contains information about the devices (the number of them in the system, computational power, communication performance, and system topology) in a way similar to SCOTCH.

To work with more than two partitions, the original Kernighan and Lin's heuristic repeatedly applies its two-partition algorithm to pairs of subsets of the partitions. This approach may fall into local minima and we avoid some of these local minima by allowing the algorithm to work with multiple partitions by considering swaps between any partitions during the whole algorithm.

The swap operation in the original Kernighan and Lin's algorithm also led to other local minima since it was limited to produce partitions with the same number of vertices of the initial partitioning. To solve this limitation, we introduced a "move" operation, in which the algorithm considers moving a single vertex from one partition to another, without requiring another vertex from the destination partition to be moved back to the source partition of the first vertex.

In the case of the communication reduction objective (or cost) function, this move operation allows all vertices to be moved to a single partition and, thus, the communication would be zero, which is the best result for this objective function. However, the dataflow graph containing the neural network model may not fit a single memory-constrained IoT device due to memory limitations. Hence, we added memory requirements for each vertex in the dataflow graph and modified the graph header to contain information about the shared parameters and biases for CNNs. Furthermore, we designed the DN$^2$PCIoT algorithm to consider the amount of memory of the devices as a restriction for the algorithm, i.e., the operations cannot be performed if there is not sufficient memory in the partitions. This feature allowed the initial partitioning and any partitioning in the middle and at the end of the algorithm to be unbalanced. At this point, unlike SCOTCH and METIS, DN$^2$PCIoT could always produce valid partitionings.

The DN$^2$PCIoT algorithm also includes a feature to factor redundant edges out of the cost computation. Redundant edges represent the transfer of the same data between partitions, which happens when there are multiple edges from one vertex to vertices that are assigned to the same partition. Neither SCOTCH nor METIS considers redundant edges, i.e., they show a number of communications that are much larger than the real value that must be indeed transferred.

Another feature that is not present in SCOTCH or METIS is the account for shared parameters and biases in the memory computation. The shared parameters are an important feature in CNNs because they can greatly reduce the amount of memory required to store the neural network. Besides that, they also help in the training phase and in avoiding overfitting as there are fewer parameters to train. A conservative solution that could be applied to SCOTCH or METIS would be to copy each set of shared parameters and biases for every vertex that needs them, however, the resultant graph would require much more memory than it is really necessary. DN$^2$PCIoT accounts for shared parameters and biases only when they are necessary, i.e., there is one corresponding set of shared parameters and biases per partition only if there is at least one neuron that needs it in the partition. This feature allows DN$^2$PCIoT to produce valid partitionings that require a realistic amount of memory and to produce valid partitionings even for very constrained devices, unlike METIS.

Finally, we designed DN$^2$PCIoT to produce partitionings that maximize the neural network inference rate or reduce the amount of transferred data per inference. Other objective functions can be easily employed in DN$^2$PCIoT due to its design. Different from METIS, which reduces the number of communications while attempting to balance the computational load and memory requirements in the hope of achieving good computational performance, DN$^2$PCIoT directly optimizes the partitioning for inference rate maximization, using the equations explained in Sections 2.2 and 2.3. In the inference rate maximization, the device or connection between devices that most limit the result is the maximum inference rate that some partitioning can provide.

The pseudocode of DN$^2$PCIoT is listed in Algorithm 1. The first step of the algorithm is to initialize *bestP*, which contains the best partitioning found so far, with the desired initial partitioning. This initial partitioning can be random-generated or defined by the user in an input file, which can be the result of another partitioning tool, for instance. It is worth noting that neither METIS nor SCOTCH can start from a partitioning obtained by another algorithm. After that, the algorithm runs in *epochs*, which are the iterations of the outer loop (Lines 3–26). This outer loop runs some epochs until the best partitioning found so far is no longer improved. For each epoch, the algorithm first unlocks all the vertices (Line 5) and initializes the current partitioning with the best partitioning found so far (Line 6). After that, the inner loop, which is a *step* of the epoch, searches for a better partitioning and updates the current and best partitionings (Lines 7–25). In each step, DN$^2$PCIoT seeks the best operation locally to identify which operation (swap or move) according to the objective function is better for the current partitioning (Line 8). This function is further detailed in Algorithm 2. Then, the best operation chosen in this function is applied to the current partitioning and the corresponding vertices are locked (Lines 10–15), i.e., they are not eligible to be chosen until the current epoch finishes. The best operation in each step may worsen the current partitioning because, if there are no operations that improve the partitioning, then the best operation is the one that increases the cost minimally. If there are no valid operations, the current step and the epoch finish (Lines 16–18). This happens when all vertices are locked or when there are unlocked vertices, but they cannot be moved or swapped due to memory constraints, i.e., if they are moved or swapped, then the partitioning becomes invalid. When the current partitioning is updated, its cost is compared to the best partitioning cost (Line 21) and, if the current partitioning cost is better, then the best partitioning is updated and the *bestImproved* flag is set to *true* so that the algorithm runs another epoch to attempt a better partitioning. Figure 2 shows the flowchart related to Algorithm 1 and represents a general view of our proposal (DN$^2$PCIoT).

**Algorithm 1** DN$^2$PCIoT algorithm.

```
 1: function DN$^2$PCIOT($initialPartitioning$)
 2:     $bestP \leftarrow initialPartitioning$;
 3:     repeat
 4:         $bestImproved \leftarrow false$;
 5:         $unlockAllNodes()$;
 6:         $currentP \leftarrow bestP$;
 7:         while there are unlocked nodes do
 8:             $op \leftarrow findBestValidOperation(currentP)$;
 9:             /* Perform the operation */
10:             if $op.type = SWAP$ then
11:                 $currentP.swap(op.v1, op.v2)$;
12:                 $lockVertex(op.v1)$; $lockVertex(op.v2)$;
13:             else if $op.type = MOVE$ then
14:                 $currentP.move(op.v, op.targetPartition)$;
15:                 $lockVertex(op.v)$;
16:             else if $op.type = INVALID$ then
17:                 /* No valid operations */
18:                 break;
19:             end if
20:             /* Update the best partitioning */
21:             if $currentP.cost() < bestP.cost()$ then
22:                 $bestP \leftarrow currentP$;
23:                 $bestImproved \leftarrow true$;
24:             end if
25:         end while
26:     until $bestImproved = false$
27:     return $bestP$;
28: end function
```

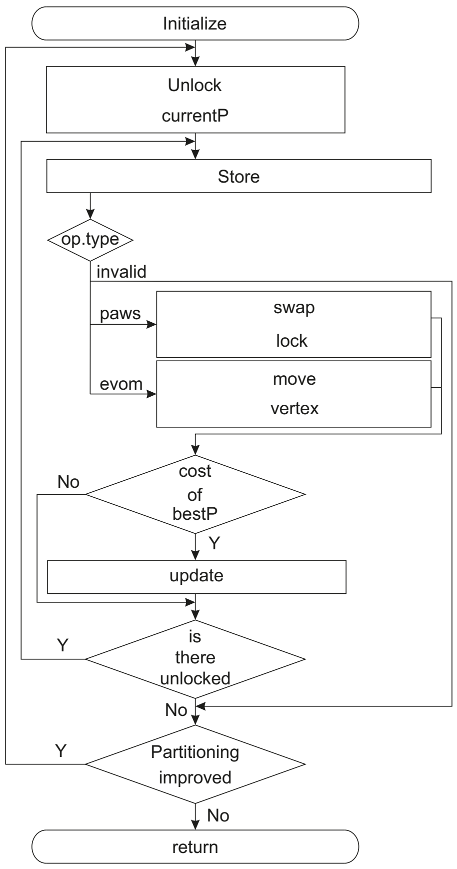


**Figure 2.** Flowchart of Algorithm 1.

Algorithm 2 shows the pseudocode for the *findBestValidOperation()* function. First, the algorithm initializes the *op* type with "invalid". If this function returns this value, then there are no operations that maintain the partitioning valid. After that, a loop runs through all the unlocked vertices searching for the best valid operation for each vertex in this set (Lines 3–32). For each vertex, the algorithm searches for the best move for it (Lines 4–15) and the best swap using this vertex (Lines 16–31). In the best move search, a loop runs through all the partitions (Lines 5–15). In this loop, the algorithm changes the current partition of the vertex being analyzed (Line 6), checks if the partitioning remains valid (Line 7), calculates the new cost of this partitioning according to the objective function (Line 8), checks if this new partitioning has a better cost than the current one (larger inference rate or fewer communications) or if no valid operation was found so far (Line 9), and updates, if necessary, *bestCost* with the better value and *op* with the move operation and the corresponding vertex and partition (Lines 10–12). In the best swap search, another loop runs through all the unlocked vertices (Lines 16–31). In this loop, the algorithm changes the current partition of both vertices that are being analyzed (Lines 17–19), checks if the partitioning remains valid (Line 20), calculates the new cost of this partitioning according to the objective function (Line 21), checks if this new partitioning has a better cost than the current one (larger inference rate or fewer communications) or if no valid operation was found so far (Line 22), and updates, if necessary, *bestCost* with the better value and *op* with the swap operation and the corresponding vertices and partitions (Lines 23–25). At the end of the loop, the original partitions of the vertices being analyzed are restored to proceed with the swap search (Lines 28–29). After the outer loop finishes, the best operation found in this function is returned to DN$^2$PCIoT (or the "invalid" operation, if no valid operations were found).

**Algorithm 2** *findBestValidOperation* function.

```
function FINDBESTVALIDOPERATION($currentP$)
    $op.type \leftarrow INVALID$;
    for $i \leftarrow unlocked.first$ to $unlocked.last$ do
        $originalPi \leftarrow currentP[i]$;
        for $p \leftarrow 1$ to $numberOfPartitions$ do
            $currentP[i] \leftarrow p$;
            if $validPartitioning(currentP) = true$ then
                $cost \leftarrow computeCost(currentP)$;
                if $cost < currentP.cost$ or $op.type = INVALID$ then
                    $bestCost \leftarrow cost$;
                    $op \leftarrow moveOp(i, p)$;
                    $op.type \leftarrow MOVE$;
                end if
            end if
        end for
        for $j \leftarrow unlocked.first$ to $unlocked.last$ do
            $originalPj \leftarrow currentP[j]$;
            $currentP[i] \leftarrow originalPj$;
            $currentP[j] \leftarrow originalPi$;
            if $validPartitioning(currentP) = true$ then
                $cost \leftarrow computeCost(currentP)$;
                if $cost < currentP.cost$ or $op.type = INVALID$ then
                    $bestCost \leftarrow cost$;
                    $op \leftarrow swapOp(i, originalPi, j, originalPj)$;
                    $op.type \leftarrow SWAP$;
                end if
                /* Restore current partitioning */
                $currentP[i] \leftarrow originalPi$;
                $currentP[j] \leftarrow originalPj$;
            end if
        end for
    end for
    return $op$;
end function
```

## 4. Methodology

In this section, we show the LeNet models and the device characteristics that we used in the experiments as well as the experiment details and approaches.

### *4.1. LeNet Neural Network Model*

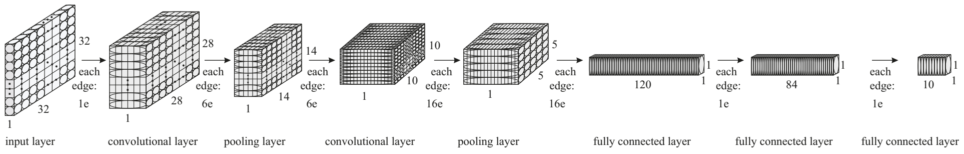
In this work, we used the original LeNet-5 DNN architecture [27] as a case study. Although LeNet is the first successful CNN, its lightweight model is suitable for constrained IoT devices. In this paper, we show that even a lightweight model such as LeNet requires partitioning to execute on constrained IoT devices. Furthermore, several works have been recently published using LeNet [40–42], causing this CNN to be still relevant nowadays.

The LeNet neurons were grouped into vertices. The neurons in the depth dimension of the LeNet convolution and pooling layers were grouped into one vertex because two neurons in these layers in the same position of width and height but different positions in depth present the same communication pattern. Thus, a partitioning algorithm would tend to assign these vertices to the same partition. For the inference rate, this modeling only affects the number of operations that a vertex will need to calculate. In the fully connected layers, as the width and height have size one, the depth was not modeled as having size one because this would limit too much the partitioning and the constrained devices able to execute this partitioning. For instance, only one setup of our experiments would fit a partitioning with this grouping, which is the least memory-constrained setup that we used in this work.

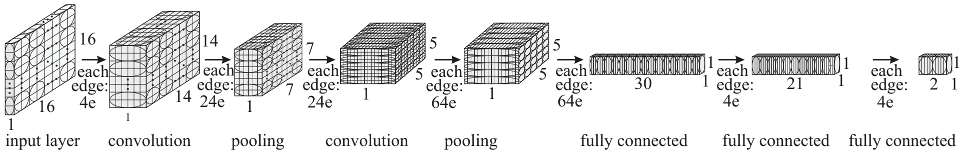
Two versions of LeNet were modeled:

- **LeNet 1:1**: the original LeNet with 2343 vertices (except for the depth explained above); and
- **LeNet 2:1**: LeNet with 604 vertices, in which the width and height of each convolution and pooling layer were divided by two, except for the last pooling layer, and the depth of the fully connected layers was divided by four, i.e., every four neurons in each of these layers were grouped to form one vertex in the model.

Figure 3 shows the dataflow graph of each LeNet version with the following per-layer data: the number of vertices in height, width, and depth, the layer type, and the amount of transferred data in byte required by each edge in each layer. In Figure 3, the cubes represent the original LeNet neurons and the circles and ellipses represent the dataflow graph vertices.



**(a)**



**(b)**

**Figure 3.** LeNet architecture and vertex granularity used in our experiments. Each cube stands for a CNN neuron while each circle is a vertex in the source dataflow graph. Edges represent data transfers and are labeled with the number of bytes per inference that each edge must transfer. (**a**) **LeNet 1:1**: the original LeNet with 2343 vertices. (**b**) **LeNet 2:1**: LeNet with 604 vertices, in which the width and height of each convolution and pooling layer were divided by two, except for the last pooling layer, and the depth of the fully connected layers was divided by four.

The grouping of the LeNet neurons reduces the dataflow graph size as we can see by the difference in the number of vertices for each graph. This reduction decreases the partitioning execution time so that we can perform more experiments in a shorter time frame. LeNet 1:1 is a more fine-grained model, thus, it may achieve better results than a less fine-grained model such as LeNet 2:1. We are aware that this approach constrains the partitioning algorithm because it cannot assign vertices in the original graph to different partitions since they are now grouped. However, in this work, we also want to show that a coarse-grained model such as LeNet 2:1 can achieve comparable results to a fine-grained model such as LeNet 1:1 and, thus, can be employed for partitionings with adequate performance. It is also important to highlight that our approach for grouping the vertices is different from the METIS multilevel approach and we show that DN$^2$PCIoT produces better results than METIS.

Finally, Table 3 shows the number of shared parameters and biases per layer for each layer and the amount of memory and computation (the number of FLOP per inference) required by each vertex per layer per LeNet model. It is worth noting that, in the LeNet model used in this work, the pooling layers present biases and trainable coefficients. In this table and hereafter, the convolution layers are represented by *C*, the pooling layers are represented by *P*, and the fully connected layers, by *FC*.

**Table 3.** Per-layer and per-vertex characteristics of each LeNet model used in this paper.

| Characteristic | Model | Input | C1 | P1 | C2 | P2 | FC1 | FC2 | FC3 |
|---|---|---|---|---|---|---|---|---|---|
| Memory of shared parameters and biases per layer (B) | both | 0 | 1248 | 96 | 12128 | 256 | 0 | 0 | 0 |
| Memory per vertex (B) | LeNet 1:1 | 8 | 48 | 48 | 128 | 128 | 3216 | 976 | 688 |
| | LeNet 2:1 | 32 | 192 | 192 | 512 | 128 | 12864 | 3904 | 3440 |
| Computation per vertex (FLOP) | LeNet 1:1 | 0 | 306 | 36 | 765 | 96 | 51 | 240 | 168 |
| | LeNet 2:1 | 0 | 1224 | 144 | 3060 | 96 | 204 | 960 | 840 |

## *4.2. Device Characteristics*

Four different devices inspired the setups that we used in the experiments. These setups are progressively constrained in memory, computational power, and communication performance and these values are shown in Table 4. The first column shows the maximum number of devices allowed to be used in each experiment. The second column shows the name of the device model that inspired each experiment. The third column shows the amount of Random Access Memory (RAM) that each device provides, which is available in the respective device datasheet [43–46]. The amount of RAM that each device provides varies from 16 KiB to 388 KiB (1 Kibibyte (KiB) = 1024 B). The fourth column represents the estimated computational performance of each device, which varies from 1.6 MFLOP/s to 180 MFLOP/s. Finally, communication is performed through a wireless medium. As this medium is shared with all the devices, the communication performance is an estimation that depends on the number of devices used in the experiments. Therefore, considering connections able to transfer up to 300 Mbits/s, the communication performance for each device varies from 9.4 KiB/s to 6103.5 KiB/s.

The reasoning for the maximum number of devices allowed to participate in the partitioning is the following. As the amount of memory provided by each device decreases, we need to employ more devices to enable a valid partitioning. Furthermore, the memory of shared parameters and biases should be taken into account when choosing the number of devices in an experiment because of the experiments that start with random-generated partitionings. To make these experiments work, each device should be able to contain at least one vertex of each neural network layer and its respective shared parameters and biases. This condition, in some cases, may increase the number of needed devices. For instance, the memory needed for LeNet (to store intermediate results, parameters, and biases) is 546.625 KiB if each layer is entirely assigned to one device. If the devices provide up to 64 KiB, it is possible to achieve valid partitionings using nine devices to fit the LeNet model. However, to start with random-generated partitionings and, thus, requiring that each device can contain at least one vertex of each layer and its respective shared parameters and biases, the number of required devices increases to 11 to produce valid random-generated partitionings.

**Table 4.** Device data and the maximum number of devices allowed to be used in the experiments.

| Number of Devices Allowed to Be Used in the Experiments | Device Model | Device Amount of RAM (KiB) | Device Estimated Computational Power (FLOP/s) | Communication Performance between Each Device (KiB/s) |
|---|---|---|---|---|
| 2 | STM32F469xx [43] | 388 | $180 \times 10^6$ | 6103.5 |
| 4 | Atmel SAM G55G [44] | 176 | $120 \times 10^6$ | 3051.8 |
| 11 | STM32L433 [45] | 64 | $80 \times 10^6$ | 332.9 |
| 56 | STM32L151VB [46] | 16 | $1.6 \times 10^6$ | 11.9 |
| 63 | STM32L151VB [46] | 16 | $1.6 \times 10^6$ | 9.4 |

For each experiment, the communication links between each device present the same performance, which is constant during the whole partitioning algorithm. The difference in the communication performance for the most constrained setups (with 56 and 63 devices) is due to the different number of devices sharing the same wireless connection. Thus, for the experiment in which the system can employ up to 63 devices for the partitioning, the communication links perform a little worse than with up to 56 devices, although the same device models with the same available memory and computational power are used.

### *4.3. Types of Experiments*

For each setup in Table 4, two experiments were performed:

- the **free-input-layer** experiment, in which all the LeNet model vertices were free to be swapped or moved; and
- the **locked-input-layer** experiment, in which the LeNet input layer vertices were initially assigned to the same device and, then, they were locked, i.e., the input layer vertices could not be swapped or moved during the whole algorithm.

The free-input-layer experiments allow all the vertices to freely move from one partition to the others, including the input layer vertices. These experiments represent situations in which the device that produces the input data is not able to process any part of the neural network and, thus, must send its data to nearby devices. In this case, we would have to add more communication to send the input data (the LeNet input layer) from the device that contains these data to the devices chosen by the approaches in this work. However, as the increased amount of transferred data involved in sending the input data to nearby devices is fixed, it does not need to be shown here. On the other hand, the locked-input-layer experiments represent situations in which the device that produces the input data can also perform some processing, therefore, no additional cost is involved in this case.

Nine partitioning approaches were employed for each experiment listed in this subsection (for each setup and free and locked inputs). These approaches are explained in the next subsections and the corresponding visual partitionings are shown for the approaches that cause it to be necessary. It is worth noting that these visual partitionings are not considered the results of this paper and are shown here for clarification of the approaches.

### *4.4. Per Layers: User-Made per-Layer Partitioning (Equivalent to Popular Machine Learning Frameworks)*

The first approach to performing the experiments is the per-layer partitioning performed by the user. In this approach, the partitioning is performed per layers, i.e., a whole layer should be assigned to a device. This partitioning is offered by popular ML tools such as TensorFlow, DIANNE, and DeepX. TensorFlow allows a fine-grained partitioning, but only if the user does not use its implemented functions for each neural network layer type.

Considering the LeNet model [27] used in both versions of our experiments, it is possible to calculate the layer that requires the largest amount of memory. This layer is the second fully connected layer (the last but one LeNet layer), which requires 376.875 KiB for the parameters, the biases, and to store the layer final result. Thus, when considering the constrained devices chosen for our experiments (Table 4), it is possible to see that there is only one setup that is capable of providing the necessary amount of memory that a LeNet per-layer partitioning requires. This setup is the least constrained in our experiments and allows a maximum of two devices to be employed in the partitioning.

In the per-layer partitioning approach, the partitioning is performed by the user, so we partitioned LeNet for the first setup and show the resultant partitioning in Figure 4. In this figure, only the partitioning for LeNet 2:1 is shown because the partitioning for LeNet 1:1 is equivalent. It is worth noting that each color in this figure and in all the figures that represent visual partitionings corresponds to a different partition.

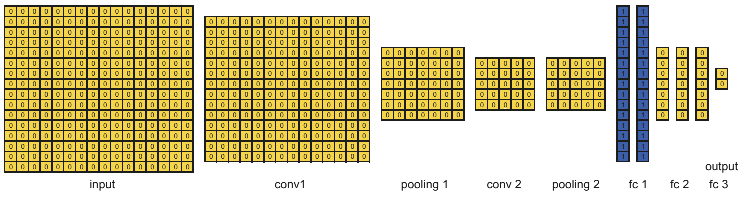


**Figure 4.** LeNet 2:1 per-layer partitioning provided by the user.

### 4.5. Greedy: A Greedy Algorithm for Communication Reduction

The second approach is a simple algorithm that aims to reduce communication. In this algorithm, whose pseudocode is listed in Algorithm 3, the layers are assigned to the same device in order until it has no memory to fit some layer. Next, if there is any space left in the device and the layer type is convolution, pooling, or input, then a two-dimensional number of vertices (width and height) that fit the rest of the memory of this device are assigned to it or, if the layer is fully connected, then a number of vertices that fit the rest of the memory of this device are assigned to it. After that or if there is any space left in the device, the next layer or the rest of the current layer is assigned to the next device and the process goes on until all the vertices are assigned to a device. It is worth noting that Algorithm 3 assumes that there is a sufficient amount of memory provided by the setups for the neural network model. Furthermore, this algorithm can partition graphs using fewer devices than the total number of devices provided. This algorithm contains two loops that depend on the number of layers (L) and the number of devices (D) of the setup, which render the algorithm complexity equals to O(L+D). However, it is worth noticing that both L and D are usually much smaller than the number of neurons of the neural network.

Figure 5 shows the visual partitioning using the greedy algorithm for each experiment. It is worth noting that this algorithm works both for the free-input-layer and the locked-input-layer experiments because the input layer could be entirely assigned to the same device for all setups. Furthermore, as the partitioning scheme is similar for LeNet 2:1 and LeNet 1:1 in the experiments with 2, 4, and 11 devices, only the partitionings for LeNet 2:1 are shown in Figure 5a–c for the sake of simplicity. For the experiments with 56 and 63 devices, the greedy algorithm results in the same partitioning because the same device model is employed in these experiments. However, as LeNet 2:1 employs 44 devices and LeNet 1:1 employs 38 devices, both results are shown in Figure 5d,e, respectively.

**Algorithm 3** Greedy algorithm for communication reduction.

```
1: function GREEDYALGORITHM($lenet$, $setup$)
2:     for $layer \leftarrow 1$ to $lenet.numberOfLayers$ do
3:         for $device \leftarrow setup.first$ to $setup.last$ do
4:             if $lenet[layer].memory \neq 0$ then
5:                 if $lenet[layer].memory \leq device.memory$ then
6:                     assign $layer$ to $device$;
7:                 else if $lenet[layer].type = conv$ or $pooling$ or $input$ then
8:                     assign a 2D number of vertices that fit $device$;
9:                 else if $lenet[layer].type = fully\ connected$ then
10:                    assign the number of vertices that fits $device$;
11:                end if
12:                $device.memory \leftarrow device.memory - assigned$;
13:                $lenet[layer].memory \leftarrow lenet[layer].memory - assigned$;
14:            else
15:                break;
16:            end if
17:        end for
18:    end for
19: end function
```

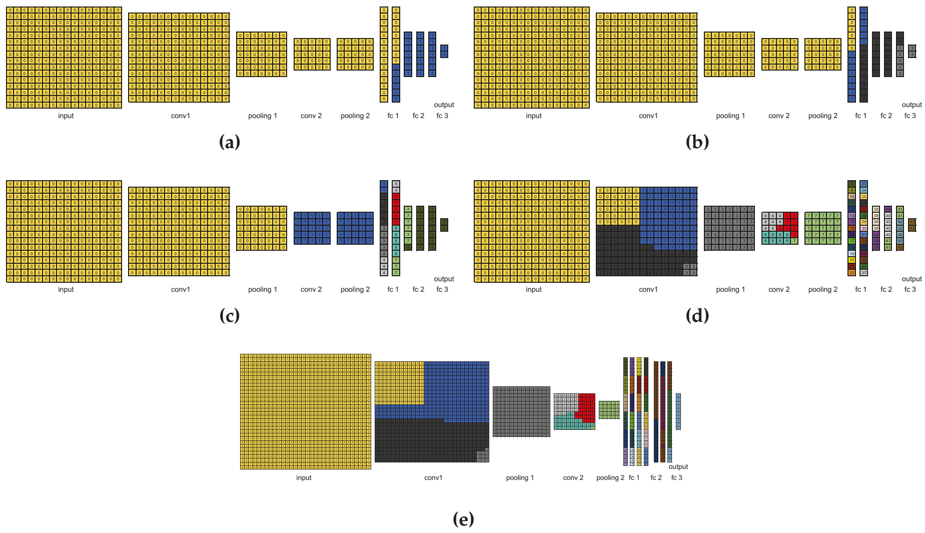


**Figure 5.** Partitionings using the greedy algorithm: (**a**) LeNet 2:1 for the two-device experiments; (**b**) LeNet 2:1 for the four-device experiments; (**c**) LeNet 2:1 for the 11-device experiments (used nine devices); (**d**) LeNet 2:1 for the 56- and 63-device experiments (used 44 devices); and (**e**) **LeNet 1:1** for the 56- and 63-device experiments (used 38 devices).

### *4.6. iRgreedy: User-Made Partitioning Aiming for Inference Rate Maximization*

The third approach is a partitioning performed by the user that aims for the inference rate maximization. The rationale behind this greedy approach is to equally distribute the vertices of each layer to each device since all the experiments present a homogenous setup. Thus, this approach employs all the devices provided for the partitioning. Besides that, again, the partition vertices are chosen in two dimensions for the input, convolution, and pooling layers.

Figure 6a shows the visual partitioning for the 11-device free-input LeNet 2:1 experiment. It is worth noting that, for the two- and four-device free-input experiments, the partitioning follows the same pattern. For the 2-, 4-, and 11-device locked-input experiments, only the input layer partitioning was changed to be assigned to only one device. Thus, these partitionings are not shown here.

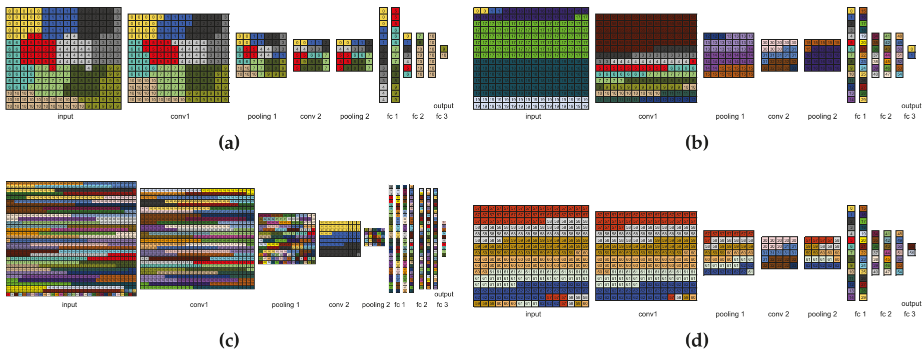


**Figure 6.** Partitionings using the inference rate greedy approach: (**a**) LeNet 2:1 for the 11-device experiments; (**b**) LeNet 2:1 for the 56-device experiments; (**c**) **LeNet 1:1** for the 56-device experiments; (**d**) LeNet 2:1 for the 63-device experiments.

For the 56- and 63-device experiments, it was not possible to employ the same rationale due to memory issues. Thus, for these experiments, the rationale was to start by the layers that require most memory and assign to the same device the largest number of vertices possible of that layer. Furthermore, in these experiments, the vertices were assigned in a per-line way because the layers were not equally distributed to all the available devices. This approach reduces the number of copies of the shared parameters and biases and, thus, allows for a valid partitioning. For the locked-input experiments, besides the input layer being changed to be entirely assigned to only one device, some adjustments had to be performed to produce valid partitionings. Figure 6b,c show the visual partitionings for the 56-device free-input LeNet 2:1 and LeNet 1:1 partitioning, respectively. Additionally, in the 63-device experiments with LeNet 2:1, the vertices in the same positions of the first layers were assigned to the same devices to reduce communication. The visual partitioning, in this case, is shown in Figure 6d. As the approach for 63 devices in LeNet 1:1 was the same for the 56 devices, the visual partitioning for 63 devices in LeNet 1:1 is not shown here either. The two approaches detailed in this subsection are greedy and, therefore, are also called inference rate greedy approach (**iRgreedy**) in the rest of this paper.

## 4.7. METIS

In this approach, the program *gpmetis* from METIS was used to automatically partition LeNet and compare the results with our approaches. The reason to choose METIS is that it is considered a widely known state-of-the-art tool used to automatically partition graphs for general purpose.

This tool offers several parameters that can be modified by the user like the number of partitions, the number of different partitionings to compute, the number of iterations for the refinement algorithms at each stage of the uncoarsening process, the maximum allowed load imbalance among the partitions, and the algorithm's objective function. The number of partitions corresponds to the maximum number of devices allowed to be employed in each setup described in Section 4.2. Thus, as METIS attempts to balance all the constraints, it always employs the maximum number of devices in each experiment. All the other parameters listed here were varied in our tests and, for our inference rate maximization objective function, the maximum allowed load imbalance among the partitions parameter was substituted for the maximum allowed load imbalance among partitions per constraint, which allows using different values for memory and the computational load. It is worth noting that, for the objective function parameter, both functions were used: *edgecut* minimization and total communication volume minimization. These parameters are detailed in the METIS manual [47].

For the locked-input experiments, the LeNet graph had the vertices from the input layer removed to run METIS with a small difference between the constraints proportion (target weights in METIS) related to the amount of memory and computational load that the input layer requires. After METIS performs the partitioning, the input layer is plugged back into the LeNet graph and we calculate the cost (inference rate or amount of transferred data) and if this partitioning is valid.

### 4.8. DN$^2$PCIoT 30R

The fifth approach that we used for the experiments was the application of DN$^2$PCIoT starting from random-generated partitionings. This approach executed DN$^2$PCIoT 30 times starting from different random-generated partitionings and we report the best value achieved in these 30 executions. It is worth noting that this approach was only executed for the LeNet 2:1 model due to the more costly execution of LeNet 1:1 starting from a random partitioning. This was the only approach that did not employ LeNet 1:1. Furthermore, DN$^2$PCIoT can discard some devices when they are not necessary, i.e., if DN$^2$PCIoT finds a better partitioning with fewer devices.

### 4.9. DN$^2$PCIoT after Approaches

The last approach corresponds to the execution of the proposed DN$^2$PCIoT starting from partitionings obtained by the other approaches considered in this work. Thus, four experiments were performed in this approach: DN$^2$PCIoT after per layers, DN$^2$PCIoT after greedy, DN$^2$PCIoT after *iRgreedy*, and DN$^2$PCIoT after METIS. This approach also allows the partitionings to employ fewer devices than the maximum number of devices allowed in each experiment. It is worth noting that no other approach in this paper can start from a partitioning obtained by another algorithm and try to improve the solution based on this initial partitioning.

## 5. Experimental Results

In this section, we show the results for all the experiments discussed in Section 4 (varying number of devices, free and locked input layer, and all the approaches) for the inference rate maximization objective function. After that, we show the pipeline parallelism factor for each setup to compare the performance of a single device to the distribution performance. Finally, the results of the inference rate maximization are plotted along with results for communication reduction to see how optimizing for one objective function affects the other. Our approaches (greedy algorithm, iRgreedy approach, DN$^2$PCIoT 30R, and DN$^2$PCIoT after the other approaches) were compared to two literature approaches: the per-layers approach (equivalent to popular ML frameworks such as TensorFlow, DIANNE, and DeepX) and METIS. We implemented DN$^2$PCIoT using C++ and executed the experiments on Linux-based operating systems.

### 5.1. Inference Rate Maximization

Table 5 shows the results for the inference rate maximization objective function for the approaches detailed in Section 4 that are compared to DN$^2$PCIoT. Table 6 shows the results for the inference rate maximization objective function for DN$^2$PCIoT 30R and DN$^2$PCIoT after all the approaches in Table 5. It is worth noting that both Tables 5 and 6 present normalized results, i.e., these results are normalized by the maximum inference rate achieved in each experiment. For instance, in the free-input two-device experiments, considering both Tables 5 and 6, the maximum inference rate was achieved by DN$^2$PCIoT after METIS with LeNet 2:1. We take this value and divide it by each result of the free-input two-device experiments. Thus, we have a value of 1.0 for the maximum inference rate in the DN$^2$PCIoT after METIS with LeNet 2:1 and the values of the other approaches reflect how many times the inference rate was worse than the maximum inference rate. In the first column of both tables, the number indicates the maximum number of devices allowed in each experiment, "free" refers to the free-input-layer experiments, and "locked" refers to the locked-input-layer experiments. For each experiment in the first column of both tables, there is a range of colors in which the red color represents the worst results while the green color represents the best results. Intermediate results are represented by yellow. As discussed in Section 4, some approaches were not able to produce valid partitionings and this is represented by an "x". For LeNet 1:1, as it is a large graph with 2343 vertices, we had to interrupt some executions and we report the best value found followed by an asterisk ("*").

**Table 5.** Normalized results for the naive approaches. The minimum and maximum consider Tables 5 and 6.

| Setup | Median of 30 Random | Per Layers 2:1 | Per Layers 2:1 | Greedy 2:1 | Greedy 1:1 | iRgreedy 2:1 | iRgreedy 1:1 | METIS 2:1 | METIS 1:1 |
|---|---|---|---|---|---|---|---|---|---|
| 2 free | 6.35 | 1.67 | 1.67 | 1.59 | 1.59 | 1.61 | 1.36 | 1.13 | 1.23 |
| 4 free | 4.09 | x | x | 2.06 | 2.06 | 1.43 | 1.21 | 1.09 | 1.14 |
| 11 free | 2.32 | x | x | 4.49 | 4.49 | 1.56 | 1.67 | 1.40 | 1.38 |
| 56 free | 2.12 | x | x | 29.25 | 29.53 | 24.00 | 1.45 | x | x |
| 63 free | 1.92 | x | x | 27.53 | 27.80 | 6.59 | 1.32 | x | x |
| 2 locked | 5.25 | 1.37 | 1.37 | 1.31 | 1.31 | 1.73 | 1.52 | 1.11 | 1.12 |
| 4 locked | 4.83 | x | x | 2.03 | 2.03 | 1.90 | 1.68 | 1.27 | 1.33 |
| 11 locked | 3.25 | x | x | 4.08 | 4.08 | 3.33 | 2.83 | 1.29 | 1.34 |
| 56 locked | 2.74 | x | x | 17.50 | 17.66 | 11.25 | 1.34 | x | x |
| 63 locked | 2.15 | x | x | 14.74 | 14.88 | 3.53 | 1.28 | x | x |

**Table 6.** Normalized results for DN$^2$PCIoT 30R and DN$^2$PCIoT after approaches.

| | | DN$^2$PCIoT after | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Setup | 30R | per Layers 2:1 | per Layers 1:1 | Greedy 2:1 | Greedy 1:1 | iRgreedy 2:1 | iRgreedy 1:1 | METIS 2:1 | METIS 1:1 |
| 2 free | 1.13 | 1.20 | 1.38 | 1.06 | 1.37 | 1.02 | 1.18 | 1.00 | 1.01 |
| 4 free | 1.38 | x | x | 1.16 | 1.25 | 1.16 | 1.14 | 1.00 | 1.01 |
| 11 free | 1.19 | x | x | 1.34 | 1.42* | 1.18 | 1.09 | 1.04 | 1.00 |
| 56 free | 1.12 | x | x | 2.62 | 5.14* | 2.12 | 1.00* | x | x |
| 63 free | 1.00 | x | x | 2.59 | 5.84* | 2.71 | 1.21* | x | x |
| 2 locked | 1.00 | 1.09 | 1.08 | 1.01 | 1.12 | 1.12 | 1.11 | 1.02 | 1.02 |
| 4 locked | 1.27 | x | x | 1.29 | 1.25 | 1.00 | 1.45 | 1.25 | 1.23 |
| 11 locked | 1.29 | x | x | 1.26 | 1.18 | 1.00 | 1.01 | 1.10 | 1.22 |
| 56 locked | 1.46 | x | x | 2.50 | 4.07* | 1.91 | 1.00* | x | x |
| 63 locked | 1.17 | x | x | 2.17 | 3.41* | 2.14 | 1.00* | x | x |

As general results, it is possible to see in Tables 5 and 6 that DN$^2$PCIoT 30R and DN$^2$PCIoT after approaches led to the best values for all the experiments. DN$^2$PCIoT 30R produced results that range from intermediate to the best results, with only 20% of the experiments yielding intermediate results. The DN$^2$PCIoT 30R results show the robustness of DN$^2$PCIoT, which can achieve reasonable results even when starting from random partitionings.

There are some important conclusions that we can draw from Table 5. The per-layer partitioning was the most limiting approach when considering constrained devices because it could only partition the model for the least constrained device setup, which used two devices. It is worth noting that this approach is the one offered by popular ML tools such as TensorFlow, DIANNE, and DeepX. Thus, we show that these tools are not able to execute DNNs in a distributed way in very constrained setups. Moreover, the per-layer partitioning produced suboptimal results for the only setup that it could produce valid partitionings. The quality of these results was due to the heavy unbalanced partitioning in the per-layer approach, which overloaded one device while a low load was assigned to the other device, as the least constrained setup offered two devices.

The state-of-the-art tool METIS also led to suboptimal results because it attempts to balance all the constraints, which are memory and computational load. Additionally, several partitionings provided by METIS were invalid because METIS does not consider a limit for the amount of memory in each partition. METIS could not produce any valid partitionings at all for the 56- and 63-device experiments because METIS cannot properly account for the memory required by the shared parameters and biases of CNNs. One way to solve this issue would be to add the memory required by the shared parameters and biases to every vertex that needs them, even if the vertices were assigned to the same partition. However, this solution would require much more memory and no partitioning using this solution would be valid for the setups used in this work. Thereby, we gave METIS the LeNet model without the memory information required by the shared parameters and biases in the hope that it would produce valid partitionings since our setups provided enough memory for LeNet and one full set of shared parameters and biases for each device. Unfortunately, METIS was not able to produce any valid partitionings in any of the 56- and 63-device constrained setups.

Finally, the greedy algorithm and the *iRgreedy* approach are simple approaches. Although they produced poor results, they could produce valid partitionings for all the proposed setups. Thus, considering the ability to produce valid partitionings, these approaches demonstrated to be better than METIS and the per-layer partitioning offered by popular ML frameworks in the proposed setups.

In Table 6, we can see that DN$^2$PCIoT starting from the partitioning produced by the other approaches also achieved results that range from intermediate to the best results. When comparing to the state-of-the-art tool METIS, DN$^2$PCIoT after METIS could improve the METIS result by up to 38%. Additionally, DN$^2$PCIoT after approaches is a better approach when compared to DN$^2$PCIoT 30R because DN$^2$PCIoT after approaches do not need to run 30 times to attempt to find the best partitioning as in the DN$^2$PCIoT 30R. Furthermore, the single execution required by DN$^2$PCIoT after each approach may run faster than DN$^2$PCIoT 30R because it starts from the intermediate result achieved by the other approaches instead of a random partitioning that usually requires several epochs to stop.

DN$^2$PCIoT after the greedy algorithm result also show the robustness of DN$^2$PCIoT because the greedy algorithm produced the worst results mostly. Nevertheless, DN$^2$PCIoT after greedy could improve the poor results of the greedy algorithm up to 11.1 times, yielding at least intermediate results in comparison to the other approaches.

The LeNet 1:1 model runs in a considerably larger time than the LeNet 2:1 model due to the difference in the number of vertices and edges of the graphs. When comparing the two LeNet models used in the experiments, it is possible to see that DN$^2$PCIoT for LeNet 2:1 led to the best result in 80% of the experiments. Thus, the results for the proposed setups suggest that it is possible to employ LeNet 2:1 for faster partitionings with limited impact on the results.

To conclude, our results show that DN$^2$PCIoT starting from 30 random-generated partitionings and DN$^2$PCIoT after the other approaches achieved the best results for inference rate maximization in all the proposed experiments and should be employed when partitioning CNNs for execution on multiple constrained IoT devices.

### 5.2. Pipeline Parallelism Factor

After showing the results for the inference rate maximization objective function, it is interesting to look at the pipeline parallelism factor to check if there is gain or loss when distributing the neural network execution. In the first column of Table 7, we have the device model and the maximum number of devices allowed to be used in each setup. The second column shows the inference rate if the entire LeNet model fit one device's memory, i.e., the inference rate based on the computational power of the devices. In this column, it is possible to see that the diminishing computational power affects the inference rate performance, as expected. In the third column, there is the best inference rate achieved in the corresponding experiments of the previous subsection. Finally, the fourth column shows the pipeline parallelism factor, which is the best inference rate achieved in the experiments (third column) divided by the inference rate if the entire LeNet fit one device's memory (second column). It is worth noting that the larger is the parallelism factor, the better, and results that are less than one indicate that there is some loss in the distribution of the neural network execution.

**Table 7.** Pipeline parallelism factor for each experiment device.

| Setups | Single Device Inference Rate * | Best Inference Rate in the Experiments | Pipeline Parallelism Factor |
|---|---|---|---|
| 2x STM32F469xx | 507.265 | 864.22 | 1.70 |
| 4x Atmel SAM G55G | 338.177 | 757.03 | 2.24 |
| 11x STM32L433 | 225.451 | 162.65 | 0.72 |
| 56x STM32L151VB | 4.509 | 21.14 | 4.69 |
| 63x STM32L151VB | 4.509 | 17.65 | 3.91 |

* In the case that the device fits the memory required by the whole LeNet model.

In Table 7, it is possible to note that there is a gain in the inference rate performance in using 2, 4, 56, and 63 devices. For the 11-device experiment, the communication among the partitions surpasses the distribution computational gain and negatively affects performance. In this case, we have 72% of the performance offered by a single device. However, we have to remember that this device cannot execute this model alone due to its memory limit. Additionally, it is worth noting that all the experiments in Table 7 were limited by the communication performance among the devices.

For the last device model, used in the 56- and 63-device experiments, we have different values for the best inference rate in the experiments due to the communication link among them, which is less powerful in the 63-device experiment and, consequently, its result is worse than for 56 devices, showing that communication impacts on the inference rate in this setup. Furthermore, the computational power of the most constrained devices used in these experiments is so low that we have gains of 4.7 and 3.9 when distributing LeNet, even considering the communication overhead. These results show that, even if we could execute LeNet in a single device, it would be more profitable to distribute the execution to achieve a higher inference rate, except for the 11-device setup.

It is important to note that, with this distribution, we enable such a constrained system to execute a CNN like LeNet. This would not be possible if only a single constrained device were employed due to the lack of memory. However, in the most constrained setups, the inference rate may be low. This can be the case, for instance, in an anomaly detection application that classifies incoming images from a camera. As most surveillance cameras generate 6–25 frames per second [48], most of the setups presented in this work satisfy the inference rate requirement for this application. Nonetheless, the most constrained setups do not satisfy the inference rate requirement of this application, thus the system may lose some frames. In the worst case, we still have 71% of the required inference rate (17.65/25), allowing the system to execute the application, even if the inference rate is not ideal.

Additional time may be required for synchronization so that a system provides correct results. The synchronization guarantees that all the data that a vertex needs to calculate its computation arrive in its inputs. Techniques such as retiming [49] can be applied to the partitionings provided by DN$^2$PCIoT to enforce synchronization. Such a technique would increase the amount of memory required to execute the CNN in a distributed form. Although this is an important issue for the deployment of CNNs on constrained IoT devices, in this work, we are not concerned by it because one of our aims is to show how better DN$^2$PCIoT can be in partitioning CNNs for constrained IoT devices when comparing to one of the state-of-the-art partitioning algorithms, which does not include synchronization overhead as well.

## 5.3. Inference Rate versus Communication

Minimizing communication is important to reduce interference in the wireless medium and to reduce the power consumed by radio operations. Common real-time applications that need to process data streams in a small period of time such as anomaly detection from camera images, for instance, the detection of vehicle crashes and robberies, may require a minimum inference rate so that there is no frame loss while reducing communication or even energy consumption is desirable so that the network is not overloaded and device energy life is augmented. On the other hand, applications that process data at a lower rate such as non-real-time image processing may require a small amount of communication so that device battery life is augmented while desirable characteristics are the network non-overload and inference rate maximization.

Thus, in this subsection, we want to show how optimizing for one of the objective functions, for instance, inference rate maximization, affects the other, for instance, communication reduction. For this purpose, Figure 7 presents the results of Section 5.1 for the inference rate maximization along with their respective values for the amount of transferred data per inference for each partitioning. We also plotted in these graphs results for the communication reduction objective function, which allow for a fair comparison in the amount of transferred data. For instance, when the objective function is the inference rate, the amount of transferred data may be larger than when the objective function is communication reduction. The inverse may also occur for the inference rate. These results were obtained by executing all the approaches discussed in Section 4, including DN$^2$PCIoT 30R and DN$^2$PCIoT after the other approaches with the communication reduction objective function.

Each graph in Figure 7 corresponds to one setup. In this figure, "comm" in the legend parentheses stands for when the approach used the communication reduction objective function, "inf" stands for the inference rate maximization objective function, "free" stands for the free-input experiment, and "locked" stands for the locked-input experiment. It is worth noting that each approach in the legend corresponds to two points in the graphs of Figure 7, one for the execution of LeNet 2:1 and one for LeNet 1:1. DN$^2$PCIoT 30R is an exception because it was executed only for LeNet 2:1, thus each approach with DN$^2$PCIoT 30R in the legend corresponds to only one point in the graphs. Another exception is the per-layer partitioning, which yielded the same result for both LeNet models and, thus, its results are represented by only one point. In this subsection, we do not distinguish the two LeNet versions employed in this work because our focus is on the approaches and so that the graphs do not get polluted.

As we want to maximize the inference rate and minimize the amount of transferred data, the best trade-offs are the ones on the right and bottom side of the graph, i.e., in the southeast position. We draw the Pareto curve [50] using the results for inference rate maximization and communication reduction achieved by all the approaches listed in Section 4 to show the best trade-offs and we divided the graphs into four quadrants considering the minimum and maximum values for each objective function. These quadrants help the visualization and show within which improvement region each approach fell.

In Figure 7a, for the two-device experiments, the Pareto curve contains two points, which correspond to the free-input DN$^2$PCIoT after METIS for the inference rate maximization and most of the locked-input DN$^2$PCIoT after approaches for communication reduction. The only approach that falls within the southeast quadrant is the free-input DN$^2$PCIoT after METIS for the inference rate maximization, which is the best trade-off between the inference rate and the amount of transferred data for this setup. Although several points fell within the southeast quadrant, it is worth noting that the three points that are closest to this best trade-off all correspond to the free-input DN$^2$PCIoT for the inference rate maximization, showing the robustness of DN$^2$PCIoT.

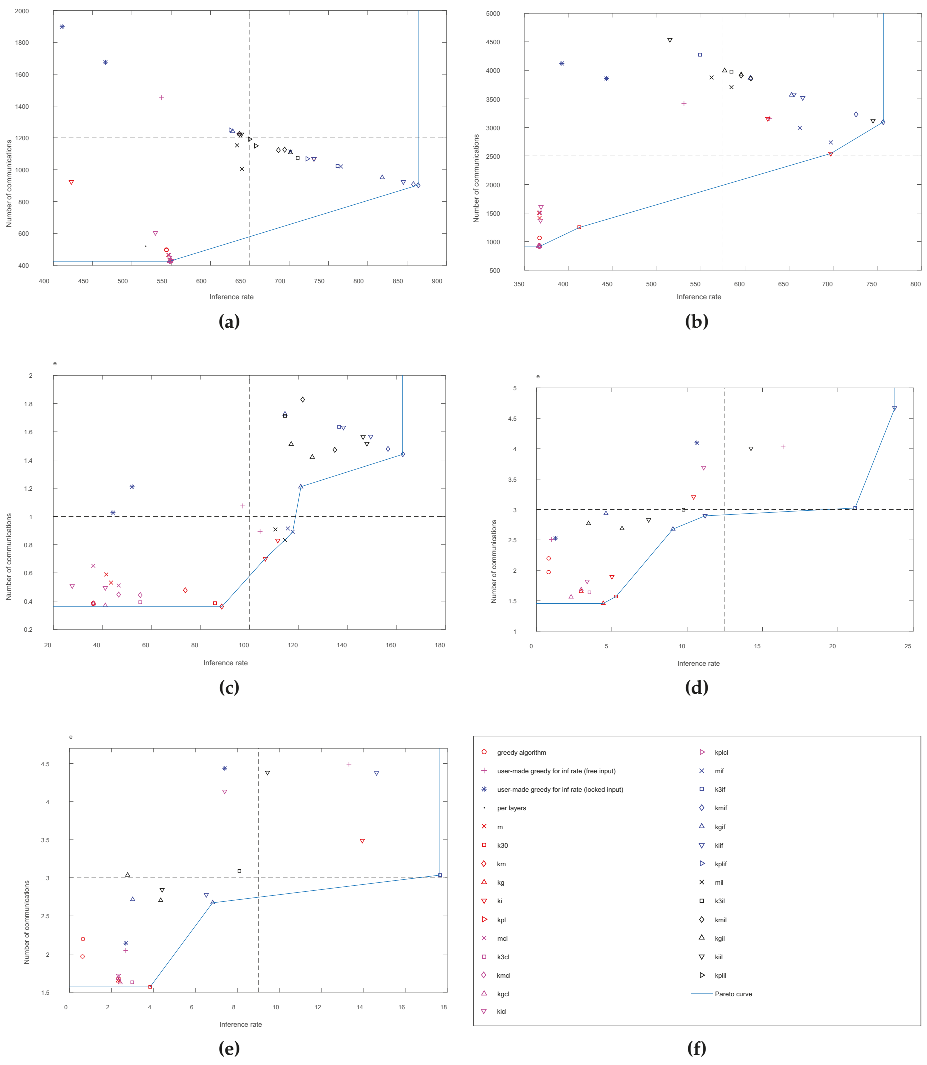


**Figure 7.** Inference rate and communication values for: (**a**) 2-device experiments; (**b**) 4-device experiments; (**c**) 11-device experiments; (**d**) 56-device experiments; and (**e**) 63-device experiments; and (**f**) legend for all graphs.

In Figure 7b, for the four-device experiments, the approach that falls both in the Pareto curve and closest to the southeast quadrant is the free-input DN$^2$PCIoT after *iRgreedy* when reducing communication. Therefore, this approach presents the best trade-off for the four-device setup.

Six points compose the Pareto curve for the 11-device experiments in Figure 7c. Three of these points falls in the best trade-off quadrant and are the free-input DN$^2$PCIoT after *iRgreedy* for communication reduction and free- and locked-input METIS for inference rate maximization. In this case, the final choice for the best trade-off depends on which condition is more important: if the application requires a larger inference rate, then METIS is the appropriate choice. On the other hand, if the application requires a smaller amount of communication, then DN$^2$PCIoT after *iRgreedy* for communication reduction is a better approach.

Six points also compose the Pareto curve for the 56-device experiments in Figure 7d. In this graph, the approach that falls both in the Pareto curve and closest to the southeast quadrant is the free-input DN$^2$PCIoT 30R when maximizing the inference rate. Therefore, this approach presents the best trade-off for the 56-device setup.

Finally, in Figure 7e, for the 63-device experiments, the approach that falls both in the Pareto curve and closest to the southeast quadrant is the free-input DN$^2$PCIoT 30R when maximizing the inference rate. This approach presents the best trade-off for the 63-device setup.

Back to the example of anomaly detection in Section 5.2, in which the application requirements involve a minimum inference rate of around 24 inferences per second while reducing communication is desirable, we can choose the best trade-offs for each setup analyzed in this subsection. In Figure 7a–c, for the setups with 2, 4, and 11 devices, respectively, all the points in the Pareto curve satisfy the application requirement of a minimum inference rate. Thus, we can choose the points that provide the minimum amount of communication. However, in Figure 7d,e, for the setups with 56 and 63 devices, respectively, the points in the Pareto curve with the minimum amount of communication do not satisfy the application requirement of the minimum inference rate. Hence, we have to choose the points with the largest inference rate in the Pareto curve of each setup, which require more communication. These results evidence the lower computational power of the devices used in the 56- and 63-device setup.

Our results suggest that our tool also deliver the best trade-offs between the inference rate and communication, with DN$^2$PCIoT providing more than 90% of the results that belong to the Pareto curve. DN$^2$PCIoT after the approaches or DN$^2$PCIoT starting from 30 random partitionings achieved the best trade-offs for the proposed setups, although these approaches only aim at one objective function. Thus, DN$^2$PCIoT 30R and DN$^2$PCIoT after approaches are adequate strategies when both communication reduction and inference rate maximization are needed, although it is possible to improve DN$^2$PCIoT with a multi-objective function containing both objectives.

### *5.4. Limitations of Our Approach*

Our algorithm presents a computational complexity of O(N$^5$), in which N is the number of vertices of the dataflow graph. Thus, the grouping of the neural network neurons may be necessary so that the algorithm executes in a feasible time. As our results suggest, the LeNet version that groups more neurons presents a limited impact on the results while the algorithms may execute faster, as the problem size is smaller. Other algorithms such as METIS performs an aggressive grouping and, thus, can execute in a feasible time. However, it is worth noting that, with 30 executions, our algorithm achieves results that are close to the best result that DN$^2$PCIoT can achieve for an experiment. On the other hand, we had to execute METIS with many different parameters to achieve valid partitionings and find the best result that METIS can get, adding up to more than 98,000 executions. Thus, METIS execution time is also not negligible.

Current CNNs such as VGG and ResNet would require more constrained devices and/or devices with a larger amount of memory so that partitioning algorithms can produce valid partitionings. However, as they are also composed of convolution, pooling, and fully connected layers, the partitioning patterns [20] tend to be similar. Additionally, as current CNNs present more neurons, strategies that groups more neurons similar to LeNet 2:1 or in multilevel partitioning algorithms such as METIS may also be required so that the partitioning algorithm executes in a feasible time.

Other strategies that we can use to reduce our algorithm execution time are to start from partitionings obtained with other tools and to interrupt execution as soon as the partitioning achieves a target value or the improvements are smaller than a specified threshold. Our algorithm can also be combined with other strategies such as the multilevel approach, which automatically groups graph vertices, but without the shortcomings of METIS, which are suboptimal values and invalid partitionings. Even with the limitations of our approach, the results suggest that there is a large space for improvements when we consider constrained devices and compare to well-known approaches.

## 6. Conclusions

In this work, we partitioned a Convolutional Neural Network for distributed inference into constrained Internet-of-Things devices using nine different approaches and we propose Deep Neural Networks Partitioning for Constrained IoT Devices (DN$^2$PCIoT), an algorithm that partitions graphs representing Deep Neural Network for distributed execution on multiple constrained IoT devices aiming for inference rate maximization or communication reduction. This algorithm adequately treats the memory required by the shared parameters and biases of CNNs so that DN$^2$PCIoT can produce valid partitionings for constrained devices. Additionally, DN$^2$PCIoT makes it easy to use other objective functions as well.

We partitioned two versions of the LeNet model with different levels of neuron grouping into five different setups aiming for inference rate maximization. Several approaches were employed for the partitionings, including the per-layer approach, which is the approach offered by popular ML tools such as TensorFlow, DIANNE, and DeepX, and the widely used tool METIS. We compared these approaches to DN$^2$PCIoT and showed that either the approaches could not produce valid partitionings for more constrained setups or they yielded suboptimal results, with DN$^2$PCIoT achieving up to 38% more inferences per second than METIS. We also calculated the inference rate for a single device of each experiment assuming the memory of this device was sufficient to execute the whole LeNet. We showed that, even if it were possible to execute the inference on a single device, there might be performance advantages of distributing its execution among multiple devices such as gains from 1.7 to 4.69 times in the inference rate provided by DN$^2$PCIoT. Finally, the results for the inference rate maximization objective function were plotted along with the respective amount of transferred data so that it was possible to see how optimizing for one objective function affects the other. Our results suggest that our tool can also deliver the best trade-offs between the inference rate and communication, with DN$^2$PCIoT providing more than 90% of the results that belong to the Pareto curve. The partitionings for both versions of LeNet achieved comparable results, with the less fine-grained LeNet model leading to the best results in 80% of the experiments. Thus, we showed that a less fine-grained model can be used in the partitionings with limited impact on the results.

**Author Contributions:** Conceptualization, F.M.C.d.O. and E.B.; Data curation, F.M.C.d.O.; Formal analysis, F.M.C.d.O. and E.B.; Funding acquisition, E.B.; Investigation, F.M.C.d.O.; Methodology, F.M.C.d.O. and E.B.; Project administration, F.M.C.d.O. and E.B.; Resources, E.B.; Software, F.M.C.d.O. and E.B.; Supervision, E.B.; Validation, F.M.C.d.O. and E.B.; Visualization, F.M.C.d.O.; Writing—original draft, F.M.C.d.O.; and Writing—review and editing, F.M.C.d.O. and E.B..

**Funding:** This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and PROCAD 2966/2014 –, by CNPq (142235/2017-2 and 313012/2017-2), FAPESP (2013/08293-7), Microsoft, and Petrobras.

**Acknowledgments:** The authors would like to thank the Multidisciplinary High Performance Computing Laboratory for its infrastructure and contributions.

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

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| B | Byte |
| C | Convolution layer |
| CNN | Convolutional Neural Network |
| DIANNE | Distributed Artificial Neural Networks for the Internet of Things |
| DN$^2$PCIoT | Deep Neural Networks Partitioning for Constrained IoT Devices |

| | |
|---|---|
| DNN | Deep Neural Network |
| FC | Fully-connected layer |
| FLOP | Floating-point operation |
| IoT | Internet of Things |
| iRgreedy | Inference rate greedy approach |
| KiB | Kibibyte |
| ML | Machine learning |
| P | Pooling layer |
| RAM | Random Access Memory |

## References

1. Vaquero, L.M.; Rodero-Merino, L. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 27–32. doi:10.1145/2677046.2677052. [CrossRef]
2. Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-Things-Based Smart Cities: Recent Advances and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 16–24. doi:10.1109/MCOM.2017.1600514. [CrossRef]
3. Cisco Systems, I. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html (accessed on 22 July 2019).
4. Miraz, M.H.; Ali, M.; Excell, P.S.; Picking, R. Internet of Nano-Things, Things and Everything: Future Growth Trends. *Future Internet* **2018**, *10*. doi:10.3390/fi10080068. [CrossRef]
5. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. doi:10.1109/JIOT.2017.2683200. [CrossRef]
6. Bormann, C.; Ersue, M.; Keranen, A. Terminology for Constrained-Node Networks. Technical report, Internet Engineering Task Force, 2014. Available online: https://doi.org/10.17487/RFC7228 (accessed on 4 April 2019).
7. Najafabadi, M.M.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*, 1. doi:10.1186/s40537-014-0007-7. [CrossRef]
8. De Coninck, E.; Verbelen, T.; Vankeirsbilck, B.; Bohez, S.; Simoens, P.; Demeester, P.; Dhoedt, B. Distributed neural networks for Internet of Things: The Big-Little approach. In Proceedings of the 2nd EAI International Conference on Software Defined Wireless Networks and Cognitive Technologies for IoT, Rome, Italy, 26–27 October 2015; pp. 1–9.
9. Grimaldi, M.; Tenace, V.; Calimera, A. Layer-Wise Compressive Training for Convolutional Neural Networks. *Future Internet* **2018**, *11*. doi:10.3390/fi11010007. [CrossRef]
10. Leroux, S.; Bohez, S.; Coninck, E.D.; Molle, P.V.; Vankeirsbilck, B.; Verbelen, T.; Simoens, P.; Dhoedt, B. Multi-fidelity deep neural networks for adaptive inference in the internet of multimedia things. *Future Gener. Comput. Syst.* **2019**, *97*, 355 – 360. doi:10.1016/j.future.2019.03.001. [CrossRef]
11. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems 28, Proceedings of the 29th Conference on Neural Information Processing Systems, Montréal, QC, Canada, 7–12 December 2015*; Cortes, C.; Lawrence, N.D.; Lee, D.D.; Sugiyama, M.; Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; pp. 1135–1143.
12. Guo, Y.; Yao, A.; Chen, Y. Dynamic Network Surgery for Efficient DNNs. In Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16), Barcelona, Spain, 5–10 December 2016; pp. 1387–1395.
13. Yao, S.; Zhao, Y.; Zhang, A.; Su, L.; Abdelzaher, T. DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys '17), Delft, The Netherlands, 5–8 November 2017; pp. 1–14.

14. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A System for Large-scale Machine Learning. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
15. De Coninck, E.; Verbelen, T.; Vankeirsbilck, B.; Bohez, S.; Leroux, S.; Simoens, P. DIANNE: Distributed Artificial Neural Networks for the Internet of Things. In Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT (M4IoT 2015), Vancouver, BC, Canada, 7–11 December 2015; pp. 19–24.
16. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Jiao, L.; Qendro, L.; Kawsar, F. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In Proceedings of the 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Vienna, Austria, 11–14 April 2016; pp. 1–12.
17. STMicroelectronics. STM32 32-bit Arm Cortex MCUs. Available online: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html (accessed on 22 July 2019).
18. Pellegrini, F. Distillating knowledge about SCOTCH. In *Combinatorial Scientific Computing, Proceedings of the Dagstuhl Seminar, Dagstuhl, Germany, 3–8 May 2009*; Naumann, U.; Schenk, O.; Simon, H.D.; Toledo, S., Eds.; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2009.
19. Karypis, G.; Kumar, V. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* **1998**, *20*, 359–392. doi:10.1137/S1064827595287997. [CrossRef]
20. De Oliveira, F.M.C.; Borin, E. Partitioning Convolutional Neural Networks for Inference on Constrained Internet-of-Things Devices. In Proceedings of the 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Lyon, France, 24–27 September 2018; pp. 266–273.
21. De Assunção, M.D.; Veith, A.S.; Buyya, R. Distributed Data Stream Processing and Edge Computing. *J. Netw. Comput. Appl.* **2018**, *103*, 1–17. doi:10.1016/j.jnca.2017.12.001. [CrossRef]
22. OpenFog Consortium Architecture Working Group. OpenFog Reference Architecture for Fog Computing. Available online: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf (accessed on 22 July 2019).
23. Zhao, H.; Zhang, W.; Sun, H.; Xue, B. Embedded Deep Learning for Ship Detection and Recognition. *Future Internet* **2019**, *11*. doi:10.3390/fi11020053. [CrossRef]
24. Venckauskas, A.; Stuikys, V.; Damaševičius, R.; Jusas, N. Modelling of Internet of Things units for estimating security-energy-performance relationships for quality of service and environment awareness. *Secur. Commun. Netw.* **2016**, *9*, 3324–3339. doi:10.1002/sec.1537. [CrossRef]
25. W, W.; Xia, X.; Wozniak, M.; Fan, X.; Damaševičius, R.; Li, Y. Multi-sink distributed power control algorithm for Cyber-physical-systems in coal mine tunnels. *Comput. Netw.* **2019**, *161*, 210–219. doi:10.1016/j.comnet.2019.04.017. [CrossRef]
26. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; ISBN 978-0262035613.
27. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc IEEE* **1998**, *86*, 2278–2324. doi:10.1109/5.726791. [CrossRef]
28. Tang, A.; Lu, K.; Wang, Y.; Huang, J.; Li, H. A Real-Time Hand Posture Recognition System Using Deep Neural Networks. *ACM Trans. Intell. Syst. Technol.* **2015**, *6*, 21:1–21:23. doi:10.1145/2735952. [CrossRef]
29. Wolf, M. Chapter 5—Program Design and Analysis. In *Computers as Components*, 4th ed.; Wolf, M., Ed.; Morgan Kaufmann: Burlington, MA, USA, 2017; pp. 221–319, ISBN 978-0-12-805387-4.
30. Benedetto, J.I.; González, L.A.; Sanabria, P.; Neyem, A.; Navón, J. Towards a practical framework for code offloading in the Internet of Things. *Future Gener. Comput. Syst.* **2019**, *92*, 424–437. doi:10.1016/j.future.2018.09.056. [CrossRef]
31. Li, H.; Ota, K.; Dong, M. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Netw.* **2018**, *32*, 96–101. doi:10.1109/MNET.2018.1700202. [CrossRef]
32. Zhao, Z.; Barijough, K.M.; Gerstlauer, A. DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2348–2359. doi:10.1109/TCAD.2018.2858384. [CrossRef]
33. Kernighan, B.W.; Lin, S. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* **1970**, *49*, 291–307. doi:10.1002/j.1538-7305.1970.tb01770.x. [CrossRef]

34. Al-Arnaout, Z.; Hart, J.; Fu, Q.; Frean, M. MP-DNA: A novel distributed replica placement heuristic for WMNs. In Proceedings of the 37th Annual IEEE Conference on Local Computer Networks, Clearwater, FL, USA, 22–25 October 2012; pp. 593–600.
35. Wen, X.; Chen, K.; Chen, Y.; Liu, Y.; Xia, Y.; Hu, C. VirtualKnotter: Online Virtual Machine Shuffling for Congestion Resolving in Virtualized Datacenter. In Proceedings of the IEEE 32nd International Conference on Distributed Computing Systems Workshop, Macau, China, 18–21 June 2012; pp. 12–21.
36. Cao, B.; Gao, X.; Chen, G.; Jin, Y. NICE: Network-aware VM Consolidation scheme for Energy Conservation in Data Centers. In Proceedings of the 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, Taiwan, 16–19 December 2014; pp. 166–173.
37. Verbelen, T.; Stevens, T.; Turck, F.D.; Dhoedt, B. Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. *Future Gener. Comput. Syst.* **2013**, *29*, 451–459. doi:10.1016/j.future.2012.07.003. [CrossRef]
38. Guerrieri, A.; Montresor, A. Distributed Edge Partitioning for Graph Processing. *arXiv* **2014**, arXiv:1403.6270v1.
39. Rahimian, F.; Payberah, A.H.; Girdzijauskas, S.; Haridi, S. Distributed Vertex-Cut Partitioning. In *Lecture Notes in Computer Science, Proceedings of the Distributed Applications and Interoperable Systems, Berlin, Germany, 3–5 June 2014*; Magoutis, K., Pietzuch, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 186–200.
40. Lopez-Jimenez, E.; Vasquez-Gomez, J.I.; Sanchez-Acevedo, M.A.; Herrera-Lozada, J.C.; Uriarte-Arcia, A.V. Columnar cactus recognition in aerial images using a deep learning approach. *Ecol. Inform.* **2019**, *52*, 131–138. doi:10.1016/j.ecoinf.2019.05.005. [CrossRef]
41. Abd Mubin, N.; Nadarajoo, E.; Shafri, H.Z.M.; Hamedianfar, A. Young and mature oil palm tree detection and counting using convolutional neural network deep learning method. *Int. J. Remote Sens.* **2019**, *40*, 7500–7515. doi:10.1080/01431161.2019.1569282. [CrossRef]
42. Ningbo, L.; Yanan, X.; Yonghua, T.; Hongwei, M.; Shuliang, W. Background classification method based on deep learning for intelligent automotive radar target detection. *Future Gener. Comput. Syst.* **2019**, *94*, 524–535. doi:10.1016/j.future.2018.11.036. [CrossRef]
43. STMicroelectronics. STM32F469xx. Available online: https://www.st.com/resource/en/datasheet/stm32f469ae.pdf (accessed on 24 July 2019).
44. Atmel. Atmel SAM G55G. Available online: http://ww1.microchip.com/downloads/en/devicedoc/Atmel-11289-32-bit-Cortex-M4-Microcontroller-SAM-G55_Datasheet.pdf (accessed on 24 July 2019).
45. STMicroelectronics. STM32L433xx. Available online: https://www.st.com/resource/en/datasheet/stm32l433cc.pdf (accessed on 24 July 2019).
46. STMicroelectronics. STM32L151x6/8/B. Available online: https://www.st.com/resource/en/datasheet/stm32l151vb.pdf (accessed on 24 July 2019).
47. Karypis, G. METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 5.1.0. Available online: http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf (accessed on 30 March 2019).
48. Honovich, J. Frame Rate Guide for Video Surveillance. Available online: https://ipvm.com/reports/frame-rate-surveillance-guide (accessed on 14 July 2019).
49. Leiserson, C.E.; Saxe, J.B. Retiming synchronous circuitry. *Algorithmica* **1991**, *6*, 5–35. doi:10.1007/BF01759032. [CrossRef]
50. Kasprzak, E.; Lewis, K. Pareto analysis in multiobjective optimization using the collinearity theorem and scaling method. *Struct. Multidiscip. Optim.* **2001**, *22*, 208–218. doi:10.1007/s001580100138. [CrossRef]

*future internet*

*Article*

# An Improved Approach for Text Sentiment Classification Based on a Deep Neural Network via a Sentiment Attention Mechanism

**Wenkuan Li, Peiyu Liu *, Qiuyue Zhang and Wenfeng Liu**

School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China; 2017020847@stu.sdnu.edu.cn (W.L.); 2018309063@stu.sdnu.edu.cn (Q.Z.); liuwenfeng@stu.sdnu.edu.cn (W.L.)

* Correspondence: liupy@sdnu.edu.cn; Tel.: +86-183-6613-2394

Received: 10 February 2019; Accepted: 1 April 2019; Published: 11 April 2019

**Abstract:** Text sentiment analysis is an important but challenging task. Remarkable success has been achieved along with the wide application of deep learning methods, but deep learning methods dealing with text sentiment classification tasks cannot fully exploit sentiment linguistic knowledge, which hinders the development of text sentiment analysis. In this paper, we propose a sentiment-feature-enhanced deep neural network (SDNN) to address the problem by integrating sentiment linguistic knowledge into a deep neural network via a sentiment attention mechanism. Specifically, first we introduce a novel sentiment attention mechanism to help select the crucial sentiment-word-relevant context words by leveraging the sentiment lexicon in an attention mechanism, which bridges the gap between traditional sentiment linguistic knowledge and current popular deep learning methods. Second, we develop an improved deep neural network to extract sequential correlation information and text local features by combining bidirectional gated recurrent units with a convolutional neural network, which further enhances the ability of comprehensive text representation learning. With this design, the SDNN model can generate a powerful semantic representation of text to improve the performance of text sentiment classification tasks. Extensive experiments were conducted to evaluate the effectiveness of the proposed SDNN model on two real-world datasets with a binary-sentiment-label and a multi-sentiment-label. The experimental results demonstrated that the SDNN achieved substantially better performance than the strong competitors for text sentiment classification tasks.

**Keywords:** deep learning; sentiment attention mechanism; bidirectional gated recurrent unit; convolutional neural network

## 1. Introduction

With the exponential growth of large collections of opinion-rich resources, sentiment classification [1] has been one of the most important tasks in natural language processing (NLP), which aims to automatically classify the sentiment polarity of a given text as negative, positive or more fine-grained classes. It can help companies to process and extract precious data from massive amounts of information, which contain great business value in brand monitoring, customer services, market research, politics, and social services. For example, tracking consumers' overall appreciation of a certain product can assist merchants in adjusting their marketing strategy.

A fundamental problem of NLP is text representation learning, which is encoding the text into continuous vectors by constructing a projection from semantics to points in high-dimensional spaces. The effect of text sentiment classification mainly depends on extracting compact and informative features from unstructured text through representation learning. Mainstream representation models for text sentiment classification can be divided into two categories based on the knowledge and information they use: traditional machine learning-based representation models and current popular deep learning-based representation models. The traditional machine learning-based representation models train a sentiment classifier relying on sentiment linguistic knowledge such as bag-of-words and sentiment lexicon, where the sentiment polarity of text is largely determined to be positive if the number of positive words is larger than that of the negative ones. In contrast, the current popular deep learning-based representation models utilize the deep neural network to learn the semantic information contained in text. The performance of deep learning-based representation models is often more superior than that of machine learning-based representation models when the syntactic structure of text is complex. In this paper, we mainly focus on integrating sentiment linguistic knowledge into the deep neural network to enhance the quality of text representation learning for sentiment classification task.

The main idea of the traditional machine learning-based representation models [2] is to employ text classifiers such as the naive Bayes classifier, maximum entropy model, and support vector machines to predict the sentiment polarity of the given texts. The performance of the naive Bayes classifier for text sentiment classification is based on the conditional probability of word-level features belonging to certain sentiment class, where features are manually designed by the bag-of-words representation method [3]. The maximum entropy model is also a statistical machine learning method based on the bag-of-words representation approach and its accuracy of sentiment classification entirely depends on the quality of the hand-crafted corpus so that the process of parameter optimization is computationally intensive and time-consuming [4]. The support vector machines constructs the sentiment feature vector mainly relying on the frequency of occurrence of words in text and trains the decision function to predict the sentiment polarity of sentences [5]. The success of these machine learning algorithms generally relies on improving feature engineering work in terms of the bag-of-words representation learning method and manual sentiment lexicon. For example, considering each sentence as a vector with the following groups of features: n-grams, character n-grams, non-contiguous n-grams, POS tags, cluster n-grams, and lexicon features, support vector machines [6] can beat all strong competitors to be the best performer of traditional machine learning models in the SemEval-2014 task. However, traditional statistical representation learning-based feature engineering work is labor intensive, and it is difficult to breakthrough its performance bottleneck for text sentiment classification tasks in the current field of NLP due to the failure to encode word order and syntactic information. Although the application of sentiment linguistic knowledge in conventional machine learning approaches has reached the upper bound, this phenomenon does not mean that it is not suitable for the current popular deep learning methods. The main goal of our thesis is to explore a way to combine sentiment linguistic knowledge with deep learning methods so as to stimulate the maximum potential of the sentiment linguistic knowledge.

Recently, it has been widely acknowledged that deep learning-based representation models have achieved great success in text sentiment classification tasks [7,8]. This is because deep learning methods are capable of learning text representation from original data without laborious feature engineering work, and capture semantic relations between context words in a scalable manner better than traditional machine learning approaches. As Mikolov et al. [9] proposed, the Word2Vec method which converted each word in the text into a continuous dense vector and distributed the different syntax and semantic features of each word to each dimension in vector space, the deep learning models could apply this method to the word embedding module in order to simplify feature engineering work. Kalchbrenner et al. [10] used a convolutional neural network for modeling sentences to obtain a promising result in text sentiment classification tasks, which demonstrated n-gram features from different positions of a sentence through convolutional operations which could promote the efficiency of predicting sentiment polarity. However, the convolutional neural network completely ignores the sequence information of the text while paying attention to the local features of a sentence. In contrast to the convolutional neural network, a long short-term memory (LSTM) network [11] is good at learning sequential correlation in the text by using an adaptive gating mechanism. Tai et al. [12] verified the importance of text sequence information learned by standard LSTM to the effect of text sentiment classification and further proposed a Tree-LSTM that incorporates the linguistic syntactic structure of a sentence into this particular structure. Unfortunately, the LSTM structure has lost the ability to learn local features of text, which is an important property of the convolutional neural network. The gated recurrent unit (GRU) [13] network is an improved variant of the LSTM, which has been improved in terms of network structure and performance, but it does not change the congenital defect of LSTM in capturing the text's local features. To avoid ignoring sequence correlation information or context local features when dealing with original unstructured text, it is important to explore an effective combination strategy that takes advantage of the convolutional neural network (CNN) and the GRU network to enrich a sufficient text representation for sentiment classification in our work.

In addition, the attention mechanism is put forward to greatly improve the quality of sentiment representation learning. The seminal NLP work using the attention mechanism is neural machine translation, where different weights are assigned to source words to implicitly learn alignments for translation [14]. The core of the attention mechanism is to imitate human's attention behavior, that a human asked to read a sentence can selectively focus on parts of context words that are important for understanding the sentence. Several recent works have been proposed to design a fusion model of an attention mechanism and a deep learning method to do sentiment representation learning tasks. Zhou et al. [15] proposed a hierarchical attention model which was jointly trained with the LSTM network to capture these key sentiment signals for predicting the sentiment polarity. Zhou et al. [16] made significant progress in extracting deep meaningful sentiment features effectively by combining bidirectional LSTM with an attention mechanism. Du et al. [17] integrated the attention mechanism with a CNN to enhance the quality of extracted local text features. Our work is inspired by the characteristic of attention mechanisms. With the help of an attention mechanism as a bridge, sentiment linguistic knowledge and deep learning methods can be perfectly integrated to enhance the sentiment feature of text.

To alleviate the aforementioned limitations, in this study, we propose a sentiment-feature-enhanced deep neural network (SDNN) for text sentiment classification. First, we propose a novel sentiment attention mechanism that uses a traditional sentiment lexicon as an attention source attending to context words via the attention mechanism. Its goal is to learn more comprehensive and meaningful sentiment-aware sentence representations as input to the deep neural network, establishing an effective relationship between sentiment linguistic knowledge and deep learning methods. Second, we designed a new model based on deep learning, combining GRU and CNN, to further enhance the representation quality of textual structure information. Above all, CNN performs poorly in learning sequential correlation information, while GRU lacks the ability to extract context local features. By using our design, this model can effectively compensate for their own defects and maximize the potential of their respective strengths.

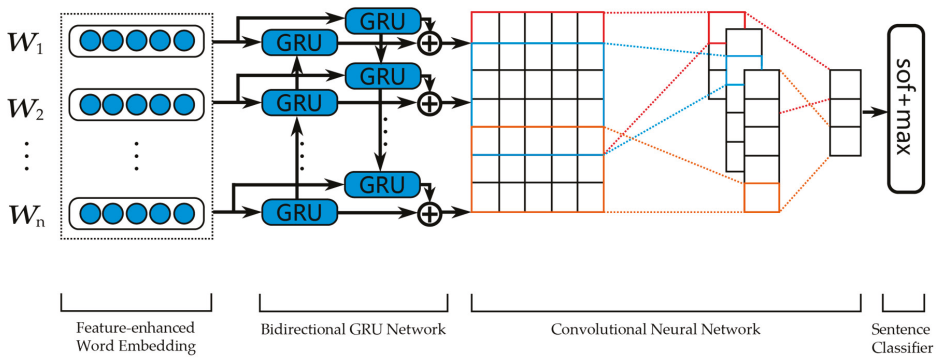
The main contributions of our work are summarized as follows:

1. We leverage traditional sentiment lexicon and a current popular attention mechanism to design a novel sentiment attention mechanism. The sentiment attention mechanism shows strong capability of interactively learning sentiment and context words to highlight the important sentiment features for text sentiment analysis.
2. Both text sequential correlation information generated by recurrent neural network and context local features captured by the convolutional neural network are essential in the text sentiment classification task, so we designed a deep neural network to effectively combine a GRU-based recurrent neural network and a convolutional neural network to enrich textual structure information.
3. Extensive experiments have been conducted on two real-world datasets with a binary-sentiment-label and a multi-sentiment-label to evaluate the effectiveness of the SDNN model for text sentiment classification. The experimental results demonstrate that the proposed SDNN model achieves substantial improvements over the compared methods.

The remainder of this paper is organized as follows. Section 2 presents the SDNN architectures for text sentiment classification in detail, including the feature-enhanced word-embedding module, bidirectional GRU network module, convolutional neural network module, and sentence classifier module. The experiment is introduced in Section 3. Section 4 presents the conclusions and discusses future work.

## 2. Model

The architecture of the proposed SDNN model is shown in Figure 1, which consists of four key components: the feature-enhanced word-embedding module, bidirectional GRU network module, convolutional neural network module, and the sentence classifier module. In addition, both the bidirectional GRU network module and the convolutional neural network module jointly constitute the deep neural network module. The goal of our model is to predict the sentiment polarity of the given sentence. In the rest of this section, we will elaborate our SDNN model in detail.



$W_1$
$W_2$
$W_n$
GRU
⊕
sof+max
Feature-enhanced Word Embedding
Bidirectional GRU Network
Convolutional Neural Network
Sentence Classifier

**Figure 1.** The architecture of the proposed sentiment-feature-enhanced deep neural network (SDNN) model.

### 2.1. Feature-Enhanced Word Embedding Module

The first step is to pre-process the input sentence and sentiment resource words. In order to transfer each word in the sentence to a real-value vector, we applied the pre-trained GloVe [18] method in the word embedding layer. Let $L \in R^{|V| \times d}$ be the embedding lookup table generated by GloVe, where $d$ is the dimension of word vector and $|V|$ is the vocabulary size. Suppose the input sentence consists of $n$ words and the sentiment resource sequence consists of $m$ words. The input sentence retrieves the word vectors from $L$ and gets a list of vectors $[w_1, w_2, \cdots, w_n]$ where $w_i \in R^d$ is the word vector of the $i^{th}$ word. Similarly, the sentiment resource sequence can retrieve the word vectors from $L$ and form a list of vectors $\left[w_1^s, w_2^s, \cdots, w_m^s\right]$. In this way, we can get the matrix $W^c = [w_1, w_2, \cdots, w_n] \in R^{n \times d}$ for context words and the matrix $W^s = \left[w_1^s, w_2^s, \cdots, w_m^s\right] \in R^{m \times d}$ for sentiment resource words.

After obtaining the general matrix of the word vector, we propose the novel sentiment attention mechanism to help highlight the vital sentiment-resource-relevant context words for generating the sentence representation with enhanced sentiment features. Specifically, we leverage sentiment words in the sentiment lexicon as sentiment resource words and integrate them with the attention mechanism to emphasize more important information related to the sentiment polarity. The sentiment lexicon collects sentiment resource words from both Hu and Liu [19] and Qian et al. [20], including 10,899 sentiment resource words in total. Then, the attention mechanism uses sentiment resource words as an attention source attending to the context words to learn the feature-enhanced word embedding. In the following, we will describe the sentiment attention mechanism in detail.

First, inspired by the fact that the sentiment words can largely guide the sentiment polarity in the context of a sentence, we plan to design the word-level relationship between the sentiment words and the context words. For example, in the sentence "This movie is so wasteful of talent, it is truly disgusting", composed of the sentiment words (i.e., "wasteful" and "disgusting") and the context words (i.e., all words in this sentence except the sentiment words), "wasteful" and "disgusting" play a key role in directing the sentiment polarity of this sentence. Mathematically, we adopt the dot product operation between the context words and the sentiment words to form a correlation matrix. The specific calculation method is as follows:

$$M^s = W^c \circ (W^s)^T \tag{1}$$

where $\circ$ denotes the dot product operation and $M^s \in R^{n\times m}$ represents the relevance matrix between the context words and the sentiment words.

Then, we define the context-word-relevant sentiment word representation matrix $X^s$ generated by the dot product operation between the context words $W^c$ and the correlation matrix $M^s$:

$$X^s = (W^c)^T \circ M^s \tag{2}$$

where $X^s \in R^{d\times m}$ represents the sentiment word representation matrix related to the context words. Similarly, we can compute the sentiment-word-relevant context word representation matrix $X^c$ by the dot product between the sentiment words $W^s$ and the correlation matrix $M^s$:

$$X^c = (M^s \circ W^s)^T \tag{3}$$

where $X^c \in R^{d\times n}$ represents the context word representation matrix related to the sentiment words. The illustration of sentiment-context word correlation is shown in Figure 2.

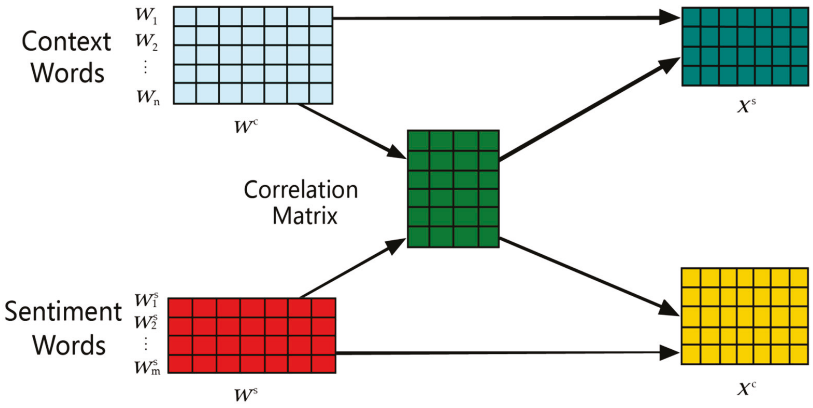


**Figure 2.** Sentiment-context word correlation.

After obtaining the sentiment-context word correlation, we adopt the attention mechanism to highlight the information contributing to predicting the sentiment polarity of the input sentence. As described in this section, we consider the sentiment influence on the context from the sentiment words, which can provide more clues to pay attention to the related sentiment features. Meanwhile, we can handle some complex situations with changing sentiment by using the attention mechanism. For example, in the sentence "It is actually my favorite kind of film, but as an adaptation, it fails from every angle", a real-world man will focus on the word "favorite" after reading the first clause of this sentence. Because "favorite" is the word that largely represents the sentiment polarity of the current sentence. Until the last word in this sentence is read, the man's attention will turn to the word "fail", which determines the sentiment polarity of the whole sentence. By using the attention mechanism that can simulate the real-world man's attention, "fails" in the context of "it fails from every angle" will be assigned more "attention" than "favorite" in the context of "my favorite kind of film". Formally, the attention score function $\beta$ is defined as follows:

$$t_s = \frac{\sum_{i=1}^{m} X_i^s}{m} \tag{4}$$

$$\beta\left(\left[X_i^c; t_s\right]\right) = u_s^T \tan\mathrm{h}\left(W_s\left[X_i^c; t_s\right]\right) \tag{5}$$

where $t_s$ denotes the overall representation of sentiment words, $u_s^T$ and $W_s$ are learnable parameters. With the attention score function $\beta$ that calculates the importance of the $i$th word $X_i^c$ in the context, the attention mechanism generates the attention vector $\alpha_i$ by:

$$\alpha_i = \frac{\exp\left(\beta\left(\left[X_i^c; t_s\right]\right)\right)}{\sum_{i=1}^{n} \exp\left(\beta\left(\left[X_i^c; t_s\right]\right)\right)} \tag{6}$$

where $\alpha_i$ represents the importance of the $i$th word in the sentence. Finally, we can attend the attention vector $\alpha_i$ to the context words:

$$x_i = \alpha_i X_i^c \tag{7}$$

where $x_i$ represents the $i$th word of the original input sentence. The final output of the feature-enhanced word-embedding layer is $X = [x_1, x_2, \cdots, x_n]$. Although the attention mechanism can highlight the sentiment features in the sentence, the model cannot fully capture the interactive information of the textual structure. In order to enhance the final representation of the sentence, we pass the sentence representation generated by the feature-enhanced word-embedding module to the deep neural network module.

## 2.2. Deep Neural Network Module

The deep neural network module is composed of two parts: Bi-GRU and CNN. As we all know, text is structured and organized. With the purpose of avoiding the destruction of the sequence structure of text, our model passes the output of feature-enhanced word-embedding layer to the Bi-GRU layer and then to the CNN layer, and obtain the final representation of the input sentence. In the following, we will detail the two parts in the order of the data flow.

As shown in Figure 1, the Bi-GRU layer contains two sub-layers for the forward and backward sequences, respectively, which is beneficial to have access to the future context as well as the past context. Since the GRU unit is a variant of LSTM, we first briefly review the LSTM for the sequence modeling task. The main idea of the LSTM is to overcome the problem of gradient vanishing and expansion in the recurrent neural network by introducing an adaptive gating mechanism that controls the data flow to and from their memory cell units. Taking the sequence vector $X = [x_1, x_2, \cdots, x_n]$ from the output of feature-enhanced word-embedding layer as an example, LSTM processes the data word-by-word corresponding to the time-step from the past to the future. At time step $t$, the current hidden state $h_t$ and the current memory-cell state $c_t$ are calculated as follows:

$$i_t = sigmoid\ (W_i \cdot [h_{t-1}, x_t] + b_i) \tag{8}$$

$$f_t = sigmoid\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{9}$$

$$o_t = sigmoid\ (W_o \cdot [h_{t-1}, x_t] + b_o) \tag{10}$$

$$\widetilde{c_t} = tanh\ (W_c \cdot [h_{t-1}, x_t] + b_c) \tag{11}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c_t} \tag{12}$$

$$h_t = o_t \odot \tan\mathrm{h}(c_t) \tag{13}$$

where $\cdot$ denotes matrix multiplication and $\odot$ stands for element-wise multiplication. $i_t$ refers to the input gate which controls the input of new information to the memory cell, $f_t$ indicates the forget gate, which controls how long certain values are held in the memory cell, and $o_t$ represents the output gate which controls how much the values stored in the memory cell affect the output activation of the block. $W_i$, $W_f$, and $W_o$ are the weight matrixes for these three gates, respectively, while $b_i$, $b_f$, $b_o$ are the bias vectors for these gates.

As for GRU, it is regarded as an optimization of LSTM, and it can not only merge the input gate $i_t$ and the forget gate $f_t$ of LSTM into the reset gate $r_t$, but also optimize the update mode of the hidden state $h_t$. In addition, the output gate $o_t$ corresponds to the update gate $z_t$. Throughout this design, the GRU can maintain the advantage of the LSTM while having a simpler structure, fewer parameters, and better convergence than the LSTM [13]. As shown in Figure 3, at each time-step $t$, the GRU transition functions are defined as follows:

$$r_t = sigmoid\ (W_r \cdot [h_{t-1}, x_t]) \tag{14}$$

$$z_t = sigmoid\ (W_z \cdot [h_{t-1}, x_t]) \tag{15}$$

$$\widetilde{h_t} = tanh\left(W_{\widetilde{h}} \cdot [r_t \odot h_{t-1}, x_t]\right) \tag{16}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t} \tag{17}$$

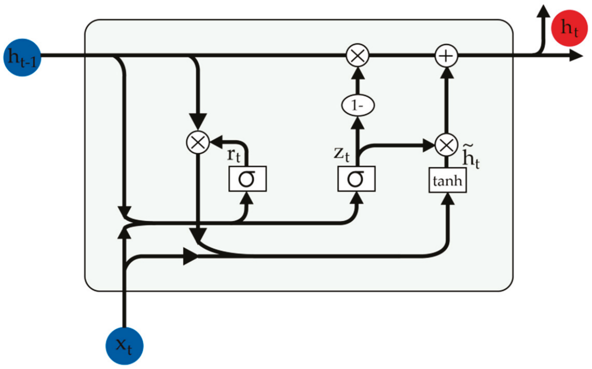


**Figure 3.** The architecture of the gated recurrent unit (GRU) cell used in the SDNN model.

In the Bi-GRU layer, the forward GRU layer processes the sentence word-by-word in the order of the input sequence to obtain the hidden state $\overrightarrow{h_t}$ at each time-step $t$. The backward GRU layer does the same thing, except that its input sequence is reversed. The final hidden state $h_t$ at time step $t$ is updated as:

$$\overrightarrow{h_t} = \overrightarrow{GRU}\ (x_t,\ \overrightarrow{h_{t-1}}) \tag{18}$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}\ (x_t,\ \overleftarrow{h_{t-1}}) \tag{19}$$

$$h_t = [\overrightarrow{h_t} \oplus \overleftarrow{h_t}] \tag{20}$$

where $\oplus$ denotes element-wise sum between the forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$. The output of the Bi-GRU layer is $H = [h_1, h_2, \cdots, h_n]$, where $n$ is the length of the input sentence.

Although the sentence representation obtained by the Bi-GRU layer maintains the sequence information of the sentence, it is not flexible enough to predict the sentiment polarity of the input sentence. To alleviate this problem, the SDNN model feeds the output of the Bi-GRU layer into the CNN layer. This is because the CNN has the ability of recognizing the local features inside a multi-dimensional field. Specifically, the CNN layer consists of a one-dimension convolutional layer and a max-pooling layer. First, we can treat $H = [h_1, h_2, \cdots, h_n] \in R^{n \times d}$, where $h_i$ represents the $i^{th}$ word in the sentence with $d$ dimension, as an "image" like in Figure 1, so the one-dimension convolutional layer can slide along the word dimension to convolve the matrix $H$ with multiple kernels of different widths. The convolutional operation can be represented as follows:

$$c_i = f\left(H_{i\,:i+K} \circ W_c + b_c\right) \tag{21}$$

where $\circ$ denotes the dot product operation, $K$ represents the width of convolutional kernel, $f$ is a non-linear function such as ReLU, $W_c$ is the convolutional matrix, and $b_c$ is the bias term. Each kernel corresponds to a linguistic feature detector which extracts a specific pattern of n-gram at various granularities [10]. The convolutional kernel is applied to each possible region of the matrix $H$ to produce a feature map $C = [c_1, c_2, \cdots, c_{n_K}]$ for the same width of convolutional kernel, where $n_K$ is the number of convolutional kernels. Then for each $c_i$ in $C$, the max pooling layer extracts the maximum value from the generated feature map:

$$p_i = down\left(c_i\right) \tag{22}$$

where $down(\cdot)$ represents the max pooling function. Through this way, the pooling layer can extract the local dependency within different regions to keep the most salient information, and it results in a fixed-size vector whose size is equal to $n_K$. Finally, the output of the max pooling layer with different widths is concatenated to form the final sentence representation $S^* = \left[p_1, p_2, \cdots, p_{n_K \times n_{wid}}\right]$, where $n_{wid}$ denotes the number of different width.

In the deep neural network module, the sequence information generated by Bi-GRU and the local feature captured by CNN are integrated with an effective strategy of combining Bi-GRU and CNN, which can help the sentence classifier to predict the sentiment polarity.

## 2.3. Sentence Classifier Module

For text sentiment classification, the final sentence representation $S^*$ of the input text $S$ is fed into a softmax layer to predict the probability distribution of sentence sentiment label over $C$ (number of sentiment category labels), and the sentiment label with the highest probability is selected as the final sentiment category to which the sentence belongs. The function is shown as follows:

$$\widetilde{y} = \frac{exp\left(W_o^T s^* + b_o\right)}{\sum_{i=1}^{C} exp\left(W_o^T s^* + b_o\right)} \tag{23}$$

$$y_{pre} = argmax\, \widetilde{y} \tag{24}$$

where $\widetilde{y}$ is the predicted sentiment distribution of the sentence, $y_{pre}$ is the selected sentiment label, $W_o^T$ and $b_o$ are the parameters to be learned.

In order to train the SDNN model, we adopt the categorical cross-entropy loss function as the reasonable training objective which is minimized in the training process:

$$J(\theta) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j \log \widetilde{y}_i^j + \lambda_r\Big(\sum_{\theta\,\in\,\Theta} \theta^2\Big) \tag{25}$$

where $N$ is the training set, $y$ is the one-hot distribution of the ground truth, $\lambda_r$ is the coefficient for the $L_2$ regularization, and $\Theta$ is the parameter set which includes all the parameters that need to be optimized during the training process. All the parameters are updated by the stochastic gradient descent strategy, which is defined as:

$$\Theta = \Theta - \lambda_l \frac{\partial J(\theta)}{\partial \Theta} \tag{26}$$

where $\lambda_l$ is the learning rate. The specific hyper-parameter settings are described in Section 3.2.

## 3. Experiments

### 3.1. Datasets

We conduct experiments on two publicly available datasets. The movie review (MR) dataset is a movie review dataset with one sentence per review collected by Pang and Lee [21], which aims to detect positive or negative reviews. The MR dataset has 5331 negative samples and 5331 positive samples. The Stanford Sentiment Treebank (SST) dataset is an extension of the MR by Socher et al. [22], which is manually separated from the train, development, test sets, and contains fine-grained sentiment labels (very positive, positive, neural, negative, very negative). Similar to the sample distribution of the MR dataset, the number of instances of each class in the SST dataset is approximately equal, which can effectively avoid the reduction of model generalization ability caused by the uneven distribution of dataset samples in the process of model training. In particular, since the MR dataset lacks a development set, we randomly sampled 10% of the training data as the development set. The detailed dataset statistics are shown in Table 1.

**Table 1.** The summary statistics of the two datasets. c: number of target classes, l: average sentence length, m: maximum sentence length, train/dev/test: train/development/test set size, |V|: vocabulary size, |$V_{pre}$|: number of words present in the set of pre-trained word embeddings, CV: 10-fold cross validation.

| Dataset | c | l | m | train | dev | test | \|V\| | \|$V_{pre}$\| |
|---|---|---|---|---|---|---|---|---|
| Movie review (MR) | 2 | 21 | 59 | 10,662 | - | CV | 20,191 | 16,746 |
| Stanford Sentiment Treebank (SST) | 5 | 18 | 51 | 8544 | 1101 | 2210 | 17,836 | 12,745 |

### 3.2. Implementation Details

In order to improve the quality of the dataset, we pre-processed the text data by removing stopwords (e.g., "in", "of", "from") and punctuation. Then, all word embeddings from the text data were initialized by 200-dimensional GloVe vectors pre-trained by Pennington et al. [18]. For the out-of-vocabulary words, we randomly sampled their embeddings from a uniform distribution $U$ (−0.1, 0.1). Some works adopted the fine-tuned training strategy for word vectors to improve the performance for text sentiment classification tasks [23]. In contrast, with the purpose of better reflecting the generalization ability of the model, we preferred to use the general embeddings for all datasets. What is more, we treated all the context words as sentiment resource words to implement the self-attention mechanism as if there was no sentiment resource word in the sentence.

For the deep neural network, the hidden states of the GRU unit in each layer were set to 200. In the convolution layer, we employed 1D convolutional filter windows of 3, 4, and 5 with 100 feature maps each, and a 1D pooling size of 4.

During the training process, we optimized the proposed model with the AdaDelta algorithm [24] by following the learning rate of $10^{-2}$ and the mini-batch size of 32. To alleviate the overfitting problem, we employed the dropout strategy [25], with a dropout rate of 0.5 for the Bi-GRU layer, 0.2 for the penultimate layer, and $10^{-5}$ for the coefficient $\lambda_r$ of $L_2$ regularization. To evaluate the performance of the sentiment classification task, we used Accuracy and F1 as the metrics.

### *3.3. Baseline Methods*

In order to comprehensively evaluate the performance of the SDNN, we set several strong baseline methods for sentiment classification, including traditional machine learning models, traditional deep learning models, deep learning models relying on specified parsing structure, and models extracting important information by attention mechanisms.

SVM/Feature-SVM: SVM [6] is a classic machine learning method to solve sentiment classification tasks, which won first place in the SemEval-2014 Task 4 by using the following groups of features: n-grams, character n-grams, non-contiguous n-grams, POS tags, cluster n-grams, and lexicon features. In addition, we incorporated sentiment resource words into the implementation of the SVM model (denoted as Feature-SVM).

LSTM/Bi-LSTM: LSTM/Bi-LSTM (Bidirectional Long Short-Term Memory) [12] has the capability to acquire dependencies between words in a sentence by incorporating the long short-term memory unit and the bidirectional variant into neural networks, which makes it effective in solving the problem of the long dependence of text.

CNN: A convolutional neural network [26] is a very strong baseline for text sentiment classification tasks. It can sufficiently capture the local feature information of the text to generate task-specific sentence representation.

BLSTM-C: BLSTM-C (Bi-LSTM combined with the generalized CNN) [27] combines CNN with Bi-LSTM networks in order to fuse the local information and sequence information of text to form feature-enhanced sentence representation for predicting the sentiment polarity of a sentence.

Tree-LSTM: Tree-LSTM (Tree-structured long short-term memory) [12] introduced memory information into tree-structured neural networks, relying on predefined paring structures, which helps to capture the semantic relatedness.

LR-Bi-LSTM: LR-Bi-LSTM (Linguistically Regularized-based Bidirectional Long Short-Term Memory) [20] makes use of linguistic roles with neural networks by utilizing linguistic regularization on intermediate outputs with KL divergence.

Self-Attention: Self-attention [28] can learn structured sentence embedding with a special regularization term.

### *3.4. Experimental Results*

#### 3.4.1. Overall Performance

Experimental results listed in Table 2 show that our proposed model (SDNN) performs competitively on MR and SST datasets. We can draw the following observations from Table 2.

First, we can see that the Feature-SVM outperformed the SVM on two datasets by roughly 0.9% on accuracy and 0.8% on F1. It shows that the SVM equipped with sentiment linguistic knowledge can better learn the representation of text. It also means that the sentiment linguistic knowledge is important to text sentiment classification tasks. Obviously, compared with Feature-SVM on two datasets, our SDNN model improved the accuracy by roughly 8.0% and F1 by roughly 7.3%. The main reason is that traditional machine learning-based methods pay more attention to word frequency features while ignore the context structure information.

Second, the conventional deep learning models (i.e., LSTM, Bi-LSTM, and CNN) outperformed Feature-SVM by a large margin on the MR and SST datasets. Although the performance of LSTM on the MR dataset was similar to that of the Feature-SVM, the accuracy and F1 of the LSTM on the SST dataset were improved by 4.9% and 4.3%, respectively. The LSTM was the worst performer of the conventional deep learning models. Undoubtedly, compared with the traditional deep learning models (i.e., LSTM, Bi-LSTM, and CNN), BLSTM-C had better results by effectively combining the CNN and LSTM networks, because it can simultaneously learn the sequence structure information of the text and capture the local features of the text, which helps to better understand the text structure information.

**Table 2.** Evaluation results (%) for the MR and SST datasets. The best result for each dataset is in bold. Results marked with # were retrieved from the references (i.e., [20,22,26,27]), while those unmarked with # were obtained either by our own implementation or with the same codes shared by the original authors.

| **Models** | **MR** | | **SST** | |
|---|---|---|---|---|
| | **Accuracy** | **F1** | **Accuracy** | **F1** |
| SVM | 76.4 | 78.8 | 40.7 # | 42.4 |
| Feature-SVM | 77.3 | 79.4 | 41.5 | 43.3 |
| LSTM | 77.4 # | 79.6 | 46.4 # | 47.6 |
| Bi-LSTM | 79.3 # | 81.0 | 49.1 # | 50.5 |
| CNN | 81.5 # | 82.7 | 48.0 # | 49.3 |
| BLSTM-C | 82.4 | 83.8 | 50.2 # | 52.0 |
| Tree-LSTM | 80.7 # | 82.1 | 50.1 # | 51.8 |
| LR-Bi-LSTM | 82.1 # | 83.6 | 50.6 # | 52.3 |
| Self-Attention | 81.7 | 82.9 | 48.9 | 50.1 |
| SDNN | **83.7** | **84.9** | **51.2** | **52.9** |
| SDNN w/o sentiment attention | 82.5 | 84.1 | 50.0 | 51.5 |

Further, the deep learning models with parsing structures (i.e., Tree-LSTM and LR-Bi-LSTM) outperformed the LSTM model on the two datasets by roughly 4.0% on accuracy and 3.8% on F1. These examples demonstrate that integrating external knowledge into deep neural networks can have a better understanding of the input text for sentiment analysis. As we expected, the SDNN achieved the best performance among all the strong competitors on the MR and SST datasets. Compared with the BLSTM-C, which was the best performer of all of the baseline methods on the same datasets, our SDNN model upgraded the results by about 1.2% on accuracy and 1.0% on F1. The results confirm our main idea that integrating sentiment linguistic knowledge into the deep neural network can enhance the quality of text representation learning for sentiment classification.

In order to analyze the effectiveness of the sentiment attention mechanism of SDNN, we also designed the ablation test in terms of discarding the sentiment attention mechanism (denoted as SDNN w/o sentiment attention). It can be clearly seen from the experimental results that the accuracy and F1 of the SDNN decreased sharply without regard to the sentiment attention mechanism. This proves our intuition that the proposed sentiment attention mechanism plays a crucial role in SDNN for text sentiment classification.

### 3.4.2. Effects of Different Combinations of Bi-GRU and CNN

To investigate the effectiveness of different combinations of Bi-GRU and CNN, we designed a series of models to compare the different effects of different combinations.

Bi-GRU+CNN: The Bi-GRU+CNN model proposed in the paper.

CNN+Bi-GRU: Based on Bi-GRU+CNN, the output of the word-embedding layer passes through the deep neural network in the order of the first CNN and the second Bi-GRU.

CNN-Bi-GRU: On the basis of Bi-GRU+CNN, the output of the word-embedding layer inputs into the Bi-GRU and CNN, respectively, and then the outputs of the networks are concatenated to form a representation of the sentence.

The results are shown in Table 3. We can see that the performance of CNN-Bi-GRU is mediocre relative to the other two models, but the 6.7% difference between Bi-GRU+CNN and CNN+Bi-GRU is not coincidental. The main reason is that the initial convolutional layer of CNN+Bi-GRU destroyed the sequence structure of the text so that the Bi-GRU layer behind it was just like a fully connected layer, which fails to harness the full capabilities of the Bi-GRU layer. The CNN+Bi-GRU was even worse than a regular LSTM by 0.4%. On the other hand, Bi-GRU+CNN achieved the best performance among the three variant models. This was because the initial Bi-GRU layer can encode every token in the input text into a vector that contains not only the information of the original token but also the information of the previous token. In this way, the order of each token in the generated text representation is the same as the order in the original text. Afterwards, the CNN layer will find local patterns by using the generated representation, which can further improve the accuracy. This proves that the combination of Bi-GRU and CNN is very sensitive to the improvement of deep neural network performance.

**Table 3.** Test accuracies (%) of different combinations of Bi-GRU and CNN.

| Models | MR | SST |
|---|---|---|
| Bi-GRU+CNN | 83.7 | 51.2 |
| CNN+Bi-GRU | 77.0 | 46.1 |
| CNN-Bi-GRU | 81.9 | 48.5 |

### 3.4.3. Effects of the Dimension of Sentiment Resource Words

Since the sentiment attention mechanism was added in the word-embedding layer, in order to verify the influence of different dimensions of sentiment resource words on the classification accuracy, we used SDNN to analyze the accuracy under different dimensions on the MR dataset. The experimental results are shown in Figure 4.

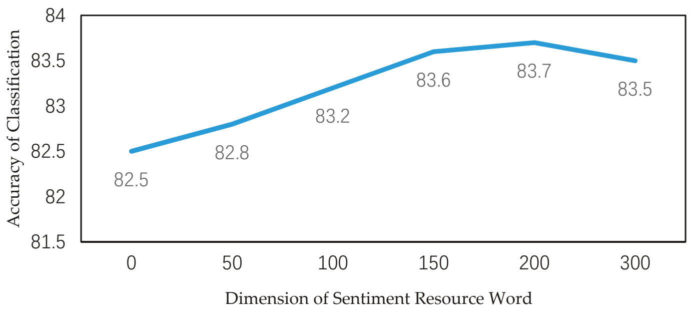


**Figure 4.** Classification of different dimensions of sentiment resource words on the MR dataset. When the dimension of the word vector is equal to 0, it means that the sentiment attention mechanism was not used in this model.

From Figure 4, we can see that when the dimension of the word vector is less than 200, the accuracy increases rapidly with the increase of the dimension. One main reason is that the model can adjust more component parameters of the sentiment resource word vector to learn the sentiment information of the sentence along with the increase of the dimension of the word vector. However, when the dimension of the word vector exceeds 200, the classification accuracy will fluctuate instead. This phenomenon is related to the meaning of the dimension of the word vector. In a word vector, each dimension represents a deep semantic feature, which is obtained by pre-training a specific corpus. Since the parameters of a word vector remain unchanged during the training process, only the parameters of the attention mechanism associated with it are optimized. In other words, when the dimension of the word vector exceeds a certain threshold, the word vector is burdened with many irrelevant parameters of vector dimension. For the optimization of the sentiment attention mechanism, these parameters are redundant, which will affect the parameter adjustment of the sentiment attention mechanism so as to affect the learning quality of sentiment linguistic knowledge. Thus, we used 200 as the dimension of the sentiment resource word vector in the experiment.

### *3.5. Visualization of Attention Mechanism*

We picked a comment sentence with changing sentiment features as a case study and visualized the attention results. To make the visualized results comprehensible, we removed the deep neural network module to make the sentiment attention mechanism directly work on the word embedding, thus we can check whether the effect of the attention mechanism conforms with human's attention performance. The visualization results are shown in Figure 5.

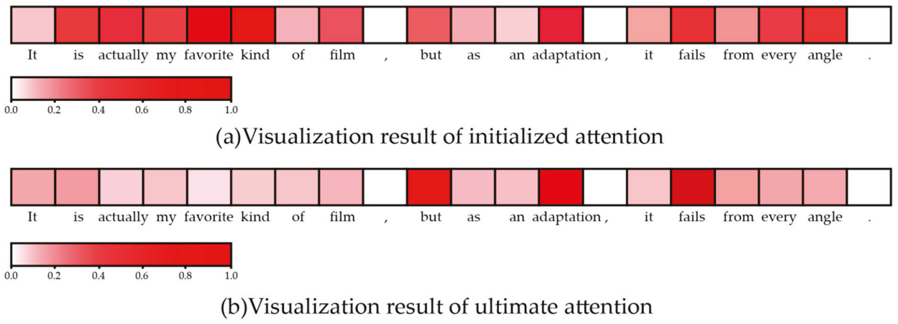


**Figure 5.** Comparison of initialized attention and ultimate attention. The attention score calculated by Equation (6) is used for the color-coding. The color depth indicates the importance degree of a word to the sentiment polarity of a sentence.

In Figure 5a, we can find that the distribution of attention weights is relatively uniform throughout the sentence when the attention mechanism is initialized. This is because the parameters of the attention mechanism are uniformly distributed during the initialization stage, which makes the attention score of each word similar. This phenomenon is similar to the performance of human's attention, when a human is asked to read a sentence, he will always give an overview of all the words. As the training process progresses, the attention mechanism gradually focuses on words that contribute a lot to the overall sentiment polarity of the sentence. This is related to the supervised learning process, in which the attention mechanism interactively learns the relationship between the context and the sentiment words to optimize the performance of text sentiment classification. In the same way, after reading a sentence, humans only pay attention to several key parts of this sentence, which helps to understand the general meaning of this sentence. As shown in Figure 5b, the ultimate result of the attention mechanism mainly highlights three words (i.e., "fails", "but", and "adaptation"), which are in line with the sentiment polarity of this sentence. Finally, with the purpose of improving the quality of word embedding, the sentiment attention mechanism is combined with the conventional word embedding, which serves as the input layer of the deep neural network to extract text structure information. From the perspective of the overall training process of the model, the sentiment attention mechanism will highlight distinguishable words contributing to the orientation of sentiment polarity, so that the model can avoid blindly learning all the context words in the next step.

## 4. Conclusions and Future Work

In this paper, we proposed the sentiment-feature-enhanced deep neural network for text sentiment classification tasks by integrating sentiment linguistic knowledge into the deep neural network via a sentiment attention mechanism. We implemented the novel sentiment attention mechanism by combining a traditional sentiment lexicon with an attention mechanism to learn sentiment-feature-enhanced word representation, bridging the knowledge gap between conventional sentiment linguistic knowledge and the deep learning method. In addition, we also designed a deep neural network to effectively combine local features within the sentence captured by the convolutional neural network and the sequence information across a sentence generated by a bidirectional GRU network, which can further improve the quality of text representation for achieving greater promotion of text sentiment classification tasks. Extensive experiments were conducted on real-world datasets with a binary-sentiment-label and a multi-sentiment-label. The experimental results showed that the proposed SDNN significantly outperformed the state-of-the-art methods for text sentiment classification tasks.

In the future, we will try to use this model to analyze the actual emotions of specific users, such as sadness, happiness or depression, which can lay a good foundation for providing better specific users' experience in terms of the marketing service industry. Furthermore, it can be seen from experimental results that the proposed model still has much room for improvement on the dataset with fine-grained sentiment labels. Therefore, the reinforcement learning method to build structured text representation without explicit linguistic structure annotations will be tried in our experiment to see if it will achieve better performance.

**Author Contributions:** W.L. (Wenkuan Li), P.L., and Q.Z. conceived and designed the study. W.L. (Wenkuan Li), Q.Z., and W.L. (Wenfeng Liu) performed the experiments. P.L. provided the data. W.L. (Wenkuan Li) and Q.Z. analyzed the data. W.L. (Wenkuan Li) and W.L. (Wenfeng Liu) proofed the algorithm. W.L. (Wenkuan Li) wrote the paper. All authors read and approved the paper.

**Acknowledgments:** This work was supported by the National Natural Science Foundation of China (Grant No. 61373148), the Science Foundation of the Ministry of Education of China (No. 14YJC860042), and the Shandong Provincial Social Science Planning Project (No. 18CXWJ01/18BJYJ04/17CHLJ33/17CHLJ30/17CHLJ18).

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

## References

1. Bing, L. Sentiment Analysis and Opinion Mining. In *Encyclopedia of Machine Learning and Data Mining*; Springer: Boston, MA, USA, 2012; Volume 30, p. 167.
2. Wang, S.; Manning, C.D. Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 8–14 July 2012; Volume 2, pp. 90–94.
3. Jiang, Q.W.; Wang, W.; Han, X. Deep feature weighting in naive Bayes for Chinese text classification. In Proceedings of the 2016 Fourth International Conference on Cloud Computing and Intelligence Systems, Beijing, China, 17–19 August 2016; pp. 160–164.
4. Yin, C.; Xi, J. Maximum entropy model for mobile text classification in cloud computing using improved information gain algorithm. *Multimed. Tools Appl.* **2016**, *76*, 1–17. [CrossRef]
5. Joachims, T. Transductive inference for text classification using support vector machines. In Proceedings of the Sixteenth International Conference on Machine Learning, Bled, Slovenia, 27–30 June 1999; pp. 200–209.
6. Kiritchenko, S.; Zhu, X.; Cherry, C.; Saif, M. Detecting aspects and sentiment in customer reviews. In Proceedings of the 8th International Workshop on Semantic Evaluation, Dublin, Ireland, 23–24 August 2014; pp. 437–442.
7. Gatt, A.; Krahmer, E. Survey of the state of the art in natural language generation: core tasks, applications and evaluation. *Vestn. Oftalmol.* **2017**, *45*, 1–16. [CrossRef]
8. Young, T.; Hazarika, D.; Poria, S. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Comput. Intell. Mag.* **2017**, *13*, 55–75. [CrossRef]
9. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; Volume 2, pp. 3111–3119.
10. Nal, K.; Edward, G.; Phil, B. A convolutional neural network for modelling sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 23–25 June 2014; pp. 655–665.
11. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

12. Tai, K.S.; Socher, R.; Manning, C.D. Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 30–31 July 2015; pp. 1556–1566.
13. Cho, K.; Merrienboer, B.V.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Proceedings of the Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1724–1735.
14. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
15. Zhou, X.J.; Wan, X.J.; Xiao, J.G. Attention-based lstm network for cross-lingual sentiment classification. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 247–256.
16. Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; Xu, B. Attention-based bidirectional long short-term memory networks for relation classification. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; p. 207.
17. Du, J.; Gui, L.; Xu, R.; He, Y. A convolutional attention model for text classification. In Proceedings of the National CCF Conference on Natural Language Processing and Chinese Computing, Dalian, China, 8–12 November 2017; pp. 183–195.
18. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
19. Hu, M.Q.; Liu, B. Mining and summarizing customer reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; pp. 168–177.
20. Qian, Q.; Huang, M.; Lei, J. Linguistically Regularized LSTMs for Sentiment Classification. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1679–1689.
21. Pang, B.; Lee, L. A Sentimental Education: Sentiment Analysis Using Subjectivity, Summarization Based on Minimum Cuts. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, 21–26 July 2004; pp. 271–278.
22. Socher, R.; Perelygin, A.; Wu, J.Y.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; Volume 1631, p. 1642.
23. Zhang, R.; Lee, H.; Radev, D. Dependency Sensitive Convolutional Neural Networks for Modeling Sentences and Documents. In Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational, San Diego, CA, USA, 12–17 June 2016; pp. 1512–1521.
24. Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M.W.; Pfau, D.; Schaul, T.; Shillingford, B.; Freitas, N.D. Learning to learn by gradient descent by gradient descent. *arXiv* **2016**, arXiv:1606.04474.
25. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
26. Kim, Y. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1746–1751.
27. Li, Y.; Wang, X.; Xu, P. Chinese Text Classification Model Based on Deep Learning. *Future Internet* **2018**, *10*, 113. [CrossRef]
28. Gui, L.; Zhou, Y.; Xu, R.; He, Y.; Lu, Q. Learning representations from heterogeneous network for sentiment classification of product re-views. *Knowl. Based Syst.* **2017**, *124*, 34–45. [CrossRef]

*future internet*

*Article*

# Dynamic Gesture Recognition Based on MEMP Network

**Xinyu Zhang * and Xiaoqiang Li**

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; xqli@i.shu.edu.cn
* Correspondence: 646599367@shu.edu.cn

Received: 12 March 2019; Accepted: 1 April 2019; Published: 3 April 2019

**Abstract:** In recent years, gesture recognition has been used in many fields, such as games, robotics and sign language recognition. Human computer interaction (HCI) has been significantly improved by the development of gesture recognition, and now gesture recognition in video is an important research direction. Because each kind of neural network structure has its limitation, we proposed a neural network with alternate fusion of 3D CNN and ConvLSTM, which we called the Multiple extraction and Multiple prediction (MEMP) network. The main feature of the MEMP network is to extract and predict the temporal and spatial feature information of gesture video multiple times, which enables us to obtain a high accuracy rate. In the experimental part, three data sets (LSA64, SKIG and Chalearn 2016) are used to verify the performance of network. Our approach achieved high accuracy on those data sets. In the LSA64, the network achieved an identification rate of 99.063%. In SKIG, this network obtained the recognition rates of 97.01% and 99.02% in the RGB part and the rgb-depth part. In Chalearn 2016, the network achieved 74.57% and 78.85% recognition rates in RGB part and rgb-depth part respectively.

**Keywords:** gesture recognition; human computer interaction; alternative fusion neural network

## 1. Introduction

Gesture communication is a widely used method in people's daily lives. Gesture interaction can be used in many kinds of scenes and has rich expressive power. For instance, sign language recognition is an important application of gestures, especially in the communication between deaf and dumb people [1]. People presently pay more and more attention to the efficiency of gesture recognition. The difficulty of gesture recognition mainly lies in the difference in body shape, video background and video noise, etc. [2]. The research of gesture recognition mainly includes static aspects and dynamic aspects. Accuracy is an important criterion to measure gesture recognition algorithms. Effective gesture recognition is still a very challenging problem [3], partly due to the cultural differences, various observation conditions, noises, relative small size of fingers in images, out-of-vocabulary motions, etc.

In traditional machine learning algorithms, HMM (Hidden Markov Model) and SVM (Support Vector Machine) are often used for the recognition of gesture [4]. SVM is often paired with HOG (Histogram of Oriented Gradient) to realize the static gesture recognition of images, which is not a suitable method for dynamic gesture recognition. In the study of dynamic gesture recognition, the HMM model is applicable to the prediction of time series, and has a good recognition rate for gestures with high complexity. However, since the HMM requires more samples to complete the graph optimization, the process of training is relatively complicated. CRF (Conditional Random Field) can be trained to recognize non-classified gesture trajectories and achieve good results [5]. However, CRF training is costly and complex, and it is not suitable for large data sets.

With the rapid development of the neural network, the recognition method has gradually transferred from traditional machine learning to deep learning. In the deep learning gesture recognition algorithm, 2D CNN and denoising self-encoder (SDAE) are often applied to feature extraction of images [6]. The 2D CNN can achieve the effect of predicting video by extracting spatial feature information of successive sets of frames in the video. 3D CNN extracts the spatio-temporal features of the entire video to get more comprehensive information [7]. Because some gesture videos may take longer time, many researchers use long short-term memory (LSTM) network to predict the gesture video [8]. LSTM can extract the timing features between frames more effectively [9]. ConvLSTM is a network structure proposed according to convolution operation and LSTM. It does not just extract spatial features like CNN, but also model according to time series like LSTM [10].

Neural networks are often used in combination when recognizing the video data set of gestures. In the combination of CNN and LSTM, videos are firstly divided into a set of frames with fixed length. Then several convolution operations and pooling operations should be carried out for each frame, and finally several feature graphs obtained after processing are predicted as input of LSTM [11]. When 3D CNN is combined with LSTM to classify video, videos should be divided into several picture sets of frames firstly, and then these frames were operated by 3D CNN in time and space according to the convolution kernel of a certain size. Finally, the processed feature set was combined to predict the input of LSTM, and we can get the accuracy [12]. However, in the combination of conventional CNN network and RNN network, single RNN operation may not obtain more accurate prediction information. Therefore, in this paper, we proposed a multi-prediction neural network with multiple mixing of 3D CNN operation and convLSTM operation. We call it the MEMP network. The MEMP network can improve the accuracy of gesture recognition. Experiments show that this network is suitable for medium and large data sets.

In the MEMP network, each gesture video needs to be split into 16 consecutive frames. These frames are subject to three consecutive 3D CNN operations and ConvLSTM. 3D CNN is used to extract spatial-temporal features of the frame set. ConvLSTM does not just predict the frame set, but also extract more spatial feature information during the prediction process. Therefore, compared with the traditional combined neural network, MEMP network retains more spatial-temporal feature information through multiple information extraction and prediction of feature maps. In dynamic gesture video, information is contained in the spatial-temporal sequence of the video. Thus, more gesture recognition accurate prediction results can be obtained. Figure 1 shows the overall structure of the network.

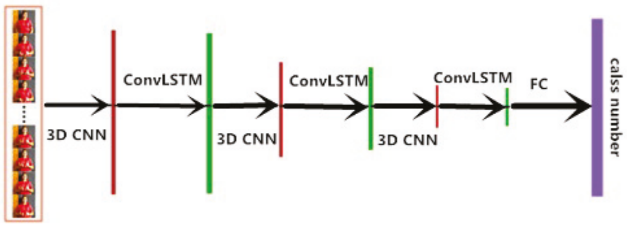


**Figure 1.** The network proposed in Figure 1. The frame set of video needs to go through three consecutive 3D CNN operations and ConvLSTM operations, and then we can get feature maps with a large amount of space-time information. Finally, feature maps are fully connected to the corresponding classification results. Each vertical line in the figure represents the state of the feature map set after a 3D CNN operation or ConvLSTM operation. The parameter transformation of each layer of the feature map is shown in Figure 2.

In the related work section, we will introduce the development of gesture recognition from traditional machine learning to deep learning. In the proposed method section, we will analyze the internal structure of the MEMP network in detail. In the experimental part, we will use three data sets to verify the characteristics of the MEMP network.

## 2. Related Work

In the study of computer vision, getting the spatial-temporal information in videos is paramount. In traditional machine learning, Ahmed and Aly used LBP and PCA for feature extraction, HMM was also used for classification [13], they have got outstanding results. However, LBP cannot distinguish between neighborhood pixels and central pixels, or neighborhood pixels are larger than central pixels. The situation may result in the loss of information extraction. Under the work of Chen and Luo, a realtime Kinect-based dynamic hand gesture recognition system which contains hand tracking, data processing, model training and gesture classification is proposed [14]. Support Vector Machine is used as the recognition algorithm in the proposed system. Methods based on the state-of-the-art handcrafted features are difficult to handle large data sets [15]. At the same time, deep neural networks have achieved remarkable results in processing large-scale data sets [15].

Neural network structure plays an important role in the study of static gesture recognition. Xing and Li proposed a CNN structure for vision-based static hand gesture recognition with competitive performance [16]. In the dynamic gesture recognition based on deep learning, the first thing to do is preprocess the gesture video. Each video is divided into a set of frames of fixed width, length and height (the frame set in this paper is $64 \times 64 \times 16$, where the frame has a length and a height of 64 and a width of 16). RNN is often used to process those frames. Chai and Liu presented an effective spotting-recognition framework based on RNN for large-scale continuous gesture recognition [17]. Naguri and Bunescu presented a gesture recognition system based on LSTM networks, the architecture was trained on raw input sequences of 3D hand positions [18]. In many new studies of gesture recognition, people tend to use 3D CNN to extract the temporal and spatial features of video [19,20]. Zhu and Zhang used the combined network of 3dcnn and convLSTM to extract the features of video, and finally conducted classification operations through SPP (Spatial Pyramid Pooling) and FC (Fully Connected Layer) [3]. However, in traditional combined neural networks, a single RNN operation may not fully grasp the spatial-temporal information. Therefore, the MEMP network obtains more decisive information by cross-using CNN and RNN.

Although 2D CNN extracts spatial features well, 3D CNN can extract more information (spatial-temporal feature information) [7]. LSTM inherits the characteristics of most RNN models and solves the problem of gradient disappearance caused by gradual reduction of gradient backpropagation process, which is often used to predict the time characteristics of sequences [9]. However, LSTM often neglects the extraction of spatial features of feature maps in the prediction of sequences, which may affect the final spatial and temporal feature information. ConvLSTM not only has the capability of LSTM time series modeling, but also can describe local features like CNN. During the work of MEMP Neural Networks, 3D CNN is used to extract the spatial and temporal feature information of each frame, and then ConvLSTM was used to predict the set of features. Repeated 3D CNN operation and ConvLSTM operation will get more distinct information. The output size is determined by the number of gestures in the data set. ReLU is the activation function of the hidden layer, and it can improve the learning speed of neural networks at different depths. This means that using ReLU activation function can avoid the vanishing gradient problem [21].

## 3. Proposed Method

In this section, we will describe the internal structure of the MEMP Neural Networks in detail. Predicted results were obtained after the video frame sets operated by three consecutive 3DCNN operations and ConvLSTM operations.

CNN plays a significant role in image feature extraction [22]. Here $F = \{f_1, f_2, .....f_{16}\}$ represents the frame set of video. The size of $f_i$ is $64 \times 64$. After a 3D CNN operation, the size of each feature map was not changed, and the number of channels increased to 8. This operation can extract spatial-temporal information of feature maps. Then, after a convLSTM operation, the set became F′ = $\left\{f'_{1,1}, f'_{1,2}, .....f'_{16,16}\right\}$. The number of channels increased to 16, the number of each channel was 16 and the size of feature map is 64 × 64. Then, through a 3D POOL operation, the number of feature maps in each channel and the length and width of each feature map were reduced by half. After similar operations of 3DCNN, convLSTM and POOL, the size of feature set became F″ = $\left\{f''_{1,1}, f''_{1,2}, .....f''_{64,4}\right\}$, with 64 channels. The number of each channel was 4, and the size of feature map was 16 × 16. After above operations, the spatial-temporal feature information of the frame set will be retained. At the end of the full connection layer, the results will be classified to obtain different types of gestures. The specific operation is as follows.

3D CNN is good at extracting features of video [23]. So in MEMP Neural Networks, 3D CNN is used to process spatial and temporal features of feature graph sets. As shown in Figure 2, there are three 3D convolution operations (C1, C2, C3), three convLSTM operations (CL1, CL2, CL3) and two pooling operations (P1, P2) in the network. The size of the three 3D convolution kernels is 2 × 2 × 2, and the step length is 1 × 1 × 1. Since the convolution mode of 'SAME' is adopted, the size of each feature map in the convolution operation will not change. The filter sizes corresponding to the three convolution operations are 8, 32 and 64 respectively. The pooled method is 'max pooling' and the pooled size is 2 × 2 × 2. After each cisterization operation, a 'droupout' layer will be followed. 'droupout' will effectively reduce the occurrence of over-fitting, which can reach the effect of regularization to a certain extent [24]. The formula of 3D CNN is as follows:

$$v_{ij}^{xyz} = ReLU(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}) \tag{1}$$

'*ReLU*' is the activation function of the hidden layer. $v_{ij}^{xyz}$ represents the current value of the coordinates (x, y, z) in the *i*-th and *j*-th feature graph sets. $b_{ij}$ represents the bias of the *i*-th layer and the *j*-th feature graph set, $w_{ijm}^{pqr}$ represents the weight of the *m*-th filter connected by the position (p, q, r) in the *i*-th layer and *j*-th feature graph set. $P_i$, $Q_i$ and $R_i$ represent the height, width and depth of the convolution kernel respectively [23].

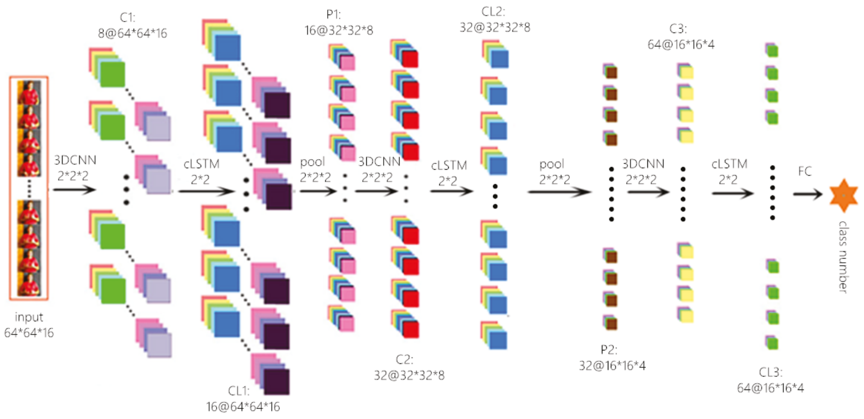


**Figure 2.** In Figure 2, the MEMP Neural Networks structure as well as the change of the feature graph in each layer of the network structure are shown. The whole process is divided into three parts, video processing, (3D CNN-convLSTM)*3 and FC. Initially, each video is split into 16 consecutive frames. Secondly, these feature sets need to be performed three convolution operations, three convLSTM operations and two pooling operations to obtain more spatial-temporal information. Finally, the network connects the full connection layer to classify these results.

Compared with traditional LSTM, convLSTM can better extract spatial and temporal features of feature graph sets [10]. The reason is that ConvLSTM can consider the spatial information of a single feature map when it processes and predicts time series events. So ConvLSTM can solve timing problems in dynamic gesture recognition more effectively. In this experiment, the processed feature graph set need to be further extracted by convLSTM after 3D CNN operation. The convolution kernel is 2 × 2, and the step lengths is 1 × 1. The convolution way is 'SAME', so the size of the feature graph is not changed. The final output filter size is 64. The main formula of convLSTM is as follows:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \tag{2}$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \tag{3}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{hc} * H_{t-1} + W_{xc} * X_t + b_c) \tag{4}$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \tag{5}$$

$$H_t = o_t \circ \tanh(C_t) \tag{6}$$

where $X_1$ ... $X_t$ are input, $C_1$ ... $C_t$ are unit output, and $H_1$ ... $H_t$ are hidden layer. $i_t$, $f_t$ and $o_t$ are three dimensional tensor of convLSTM respectively. The last two dimensions are spatial dimension (row and column). '$\circ$' is the convolution operation, and '*' is 'Hadamard product' [10]. The feature set passes through a full connection layer after a ConvLSTM operation, and the output size of the full connection layer is based on the number of gestures. The final optimization function adopts the 'Adam' algorithm. 'Adam' has high computational efficiency and low memory demand and has no impact on the gradual diagonal scaling. So it is very suitable for the problem of large data processing [24].

## 4. Experiment

### *4.1. Datasets*

This experiment used three dynamic gesture data sets: LSA (Argentinian Sign Language), chalearn 2016 (IsoGD) and SKIG.

LSA: This data set represents the Argentine sign language. The LSA data set include 3200 RGB videos and 10 non-expert subjects repeated 64 different LSA signs five times. Each video has a resolution of 1920 × 1080, 60 frames per second [25]. Table 1 shows the statistical information of training part and test part in LSA dataset.

**Table 1.** Statistical information of LSA dataset.

| | **Gesture Video** |
|---|---|
| All | 3200 |
| Training | 80% randomly |
| Testing | 20% randomly |

IsoGD: This is a large-scale gesture data set, which is derived from chalearn gesture data set [26]. The data set contains 47,933 rgb-depth video gestures made by a total of 249 gestures made by 21 different individuals. Table 2 shows the statistical information of training part and test part in IsoGD dataset.

**Table 2.** Statistical information of IsoGD dataset.

| | Gesture Video | RGB Part | Depth Part |
|---|---|---|---|
| Training | 35,878 | 35,878 | 35,878 |
| Validation | 5784 | 5784 | 5784 |
| Testing | 6271 | 6271 | 6271 |

SKIG: This data set contains 1080 rgb-depth gesture sequences collected from six individuals. All of these sequences were shot synchronously through the Kinect sensor, which includes an RGB camera and a depth camera. A total of 10 gestures were collected in this data set [27]. Table 3 shows the statistical information of training part and test part in SKIG dataset.

**Table 3.** Statistical information of SKIG dataset.

| | Gesture Video | RGB Part | Depth Part |
|---|---|---|---|
| All | 1080 | 1080 | 1080 |
| Training | 80% randomly | 80% randomly | 80% randomly |
| Testing | 20% randomly | 20% randomly | 20% randomly |

## 4.2. Video Processing

The dynamic gesture data set is generally composed of a large number of video. Video's resolution and time length will be different, so each video needs to be preprocessed. Videos will be split into a set of 16 consecutive frames with uniform time first, and then all frames need to be resized ($64 \times 64$).

## 4.3. Implementation

In terms of hardware environment, the graphics card used in our experiment is Quadro P5000 (NVIDIA, USA), CPU is E5 2650 v4 (Intel, USA), and the memory size is 128 G. Our development system is Windows 10 and the development tool used is PyCharm 2018.1 (JetBrains, Czech Republic). We use keras along with tensorflow. All data sets video are split into 16 consecutive frames of $64 \times 64$. Then 64 batches were processed when processing data, and all data were trained 100 times in total. The initial learning rate of the training process is 0.001. The exponential decay rate of the first order moment estimate is 0.9, and the exponential decay rate of the second moment estimate is 0.999. At last, the softmax function is used for the full connection classification, and the activation function is 'Adam'.

## 4.4. Experimental Results

**LSA**: 2D CNN, 3D CNN, 3D CNN + LSTM and the MEMP network structure were respectively used to train LSA data set. The network structure proposed in the paper [11] is 2D CNN + LSTM. Therefore, there are four groups of experiments in this data set, and the experimental results are shown in Table 4.

**Table 4.** LSA experimental results.

| Experiment Dataset | LSA | |
|---|---|---|
| Records | Method | Accuracy |
| 1 | 2D CNN | 93.563% |
| 2 | 2D CNN + LSTM [11] | 95.217% |
| 3 | 3D CNN | 97.892% |
| 4 | 3D CNN + LSTM | 98.451% |
| 5 | Our Method (MEMP network) | 99.063% |

As can be seen from Table 4, the MEMP neural network structure proposed in this paper has improved the recognition rate of LSA data sets by 3.846% compared with the structure proposed in this paper [11], and the MEMP network is 1.171% higher than the 3D CNN network frequently used in video processing. It can be seen that this network structure has high accuracy in processing LSA data sets.

Figure 3 show the accuracy and loss function change of LSA data set during the training of this network. The epoch of the entire network is 100. The loss function used is 'categorical-crossentropy'. The yellow line represents the verification set and the blue line represents the training set. From this we can see that when the epoch reaches 40, the accuracy and loss function of the network tends to be stable. This shows the stability of the network.

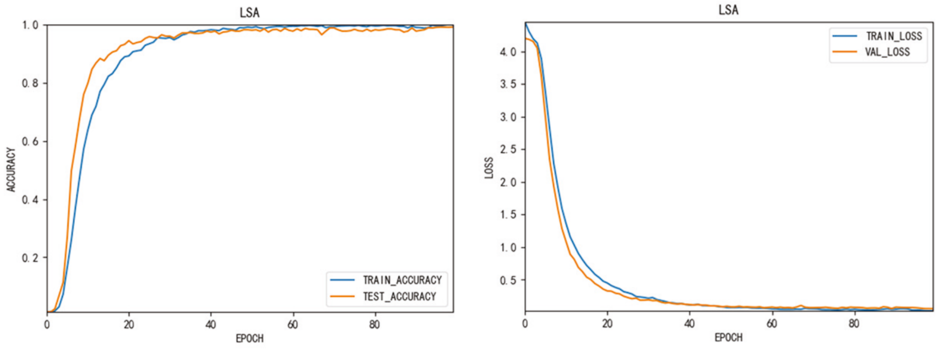


**Figure 3.** Accuracy on LSA dataset using MEMP network.

**IsoGD**: in this data set, the MEMP network and the 3D CNN network are used to train RGB and rgb-depth data sets respectively. In the paper [20], the author used C3D and LSTM networks to train IsoGD data. The experimental results are shown in Table 5.

**Table 5.** IsoGD experimental results.

| **Experiment Dataset** | **IsoGD** | |
|---|---|---|
| **Records** | **Method** | **Accuracy** |
| 1 | 2D CNN (RGB) | 46.03% |
| 2 | 2D CNN (RGB-Dep) | 49.17% |
| 3 | 3D CNN (RGB) | 49.68% |
| 4 | 3D CNN (RGB-Dep) | 55.12% |
| 5 | 3D CNN and ConvLSTM (RGB) [3] | 43.88% |
| 6 | 3D CNN and ConvLSTM (RGB-Dep) [3] | 51.02% |
| 7 | Res-C3D and Skeleton LSTM (RGB-Dep) [20] | 68.42% |
| 8 | ResNet-18 (RGB-Dep) | 66.18% |
| 9 | ResNet-34 (RGB-Dep) | 67.54% |
| 10 | 3DResNet-18 (RGB-Dep) | 71.24% |
| 11 | MEMP network (RGB) | 74.57% |
| 12 | MEMP network (RGB-Dep) | 78.85% |

It can be seen from the results that, in the IsoGD data set, the accuracy of MEMP network is 10.43% higher than that of the paper [20], which is a big breakthrough. In the same data set, MEMP network was 24.89% higher in RGB and 23.73% higher in rgb-depth data sets than the commonly used 3D CNN network. In comparison with the paper [3], we have improved 30.69% and 27.83% in RGB and RGB-Depth part respectively. These improvements illustrate the importance of multi-fetch multi-prediction for frame sets. In the experiment, the size of frame is $64 \times 64$, which is smaller than the size of paper [3] and paper [20]. This shows that MEMP network can save a lot of time. Compared with the current popular ResNet and 3D ResNet, the MEMP network also has a significant improvement in accuracy.

Figure 4 shows the variation of accuracy and loss function in training IsoGD data set. Figure 5 shows the accuracy comparison between 2D CNN, 3D CNN and our method (MEMP network) in the IsoGD (RGB-Depth) data set. From this we can see that our method in large data sets are superior to the mainstream deep learning networks in both accuracy and speed of convergence.

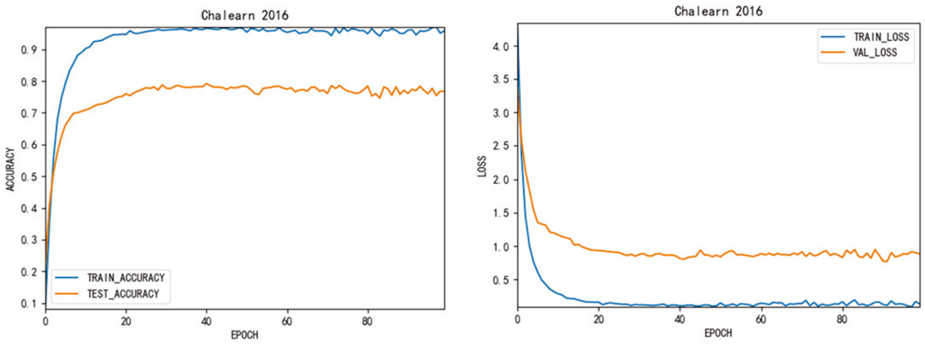


**Figure 4.** Accuracy on IsoGD dataset using MEMP network.

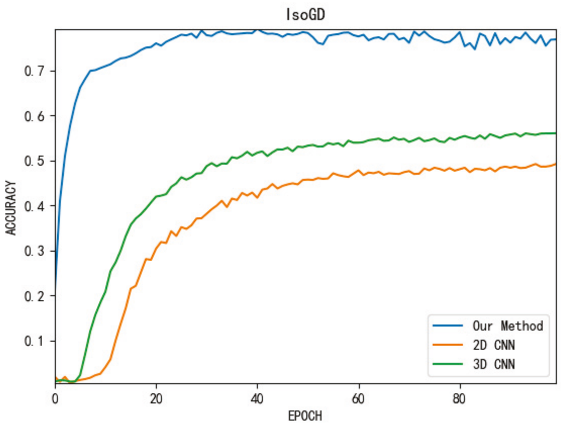


**Figure 5.** Accuracy comparison on IsoGD using mainstream method.

**SKIG**: in the paper [28], the author proposed a network structure of LPSNet to train SKIG data sets. In this experiment, the RGB data set and rgb-depth data set in SKIG were trained respectively, and the results are shown in Table 6.

**Table 6.** SKIG experimental results.

| Experiment Dataset | SKIG | |
|---|---|---|
| Records | Method | Accuracy |
| 1 | LPSNet (RGB) [28] | 96.7% |
| 2 | LPSNet (RGB-Depth) [28] | 98.7% |
| 3 | 3D CNN and ConvLSTM (RGB) [3] | 95.93% |
| 4 | 3D CNN and ConvLSTM (RGB-Dep) [3] | 98.70% |
| 5 | MEMP network (RGB) | 97.01% |
| 6 | MEMP network (RGB-Dep) | 99.02% |

As can be seen from the results in Table 6, accuracies of MEMP network in the RGB part and the rgb-depth part are higher than the LPSNet proposed in the paper [28]. Compared with the paper [3], our results are also superior to theirs.

Figure 6 shows the variation of accuracy and loss function in the training SKIG data set. As can be seen from the graph above, MEMP network has achieved outstanding results in different data sets. This indicate that our method can be applied to many medium and large data sets.

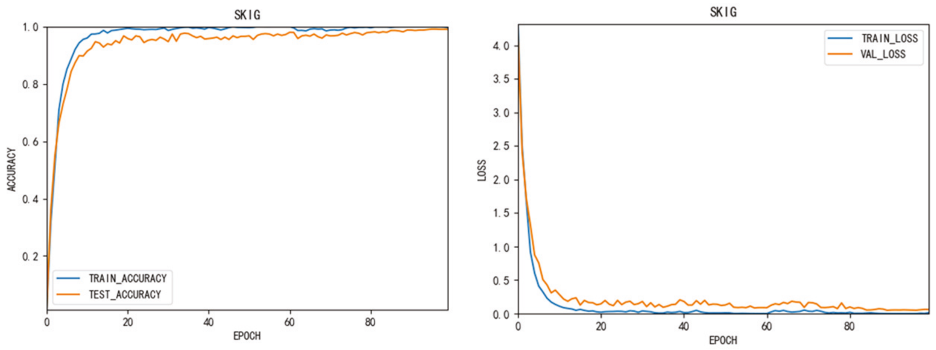


**Figure 6.** Accuracy on SKIG dataset using MEMP network.

## 5. Conclusions

Gesture recognition plays an important role both in daily life and in the direction of computer vision. The current method based on deep learning is the main research aspect of gesture recognition. In this paper, we proposed a MEMP network for gesture recognition research. The advantage of MEMP network is that 3D CNN and convLSTM are mixed several times to extract and predict the video gesture information multiple times, so as to get higher accuracy. The MEMP Neural Networks has achieved high accuracy in LSA, IsoGD and SKIG data sets, and it also indicates that this network is applicable in many medium and large video data sets. ResNet can easily realize good accuracy of image classification and location tasks. In future studies, we will use the residual network to classify gesture recognition.

**Author Contributions:** Methodology, Review, Writing and Editing, X.Z.; Validation, X.L.

**Funding:** This work is partially supported by Shanghai International Cooperation Fund Project (No. 12510708400) and Shanghai Innovation Action Plan Project (No. 16511101200) of Science and Technology Committee of Shanghai Municipality.

**Conflicts of Interest:** We have no conflict of interest.

## References

1. Nyaga, C.N.; Wario, R.D. Sign language gesture recognition through computer vision. In Proceedings of the 2018 IST-Africa Week Conference (IST-Africa), Gaborone, Botswana, 9–11 May 2018; pp. 1–8.
2. Wan, J. Results and analysis of ChaLearn LAP multi-modal isolated and continuous gesture recognition, and real versus fake expressed emotions challenges. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 3189–3197.
3. Zhu, G.; Zhang, L.; Shen, P. Multimodal gesture recognition using 3D convolution and convolutional LSTM. *IEEE Access* **2017**, *5*, 4517–4524. [CrossRef]
4. Camada, M.Y.O.; Cerqueira, J.J.F.; Lima, A.M.N. Stereotyped gesture recognition: An analysis between HMM and SVM. In Proceedings of the 2017 IEEE International Conference on Innovations in Intelligent SysTems and Applications (INISTA), Gdynia, Poland, 3–5 July 2017; pp. 328–333.
5. Ma, L.; Zhang, J.; Wang, J. Modified CRF algorithm for dynamic hand gesture recognition. In Proceedings of the 33rd Chinese Control Conference, Nanjing, China, 28–30 July 2014; pp. 4763–4767.
6. Oyedotun, O.K.; Khashman, A. Deep learning in vision-based static hand gesture recognition. *Neural Comput. Appl.* **2017**, *28*, 3941–3951. [CrossRef]
7. Du, T.; Bourdev, L.; Fergus, R. Learning spatiotemporal features with 3D convolutional networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 13–16 December 2015; pp. 4489–4497.
8. Du, T.; Ren, X.; Li, H. Gesture recognition method based on deep learning. In Proceedings of the 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), Nanjing, China, 18–20 May 2018; pp. 782–787.
9. Ese, N.; Esent, P. Long short-term memory in recurrent neural networks. *Epfl* **2001**, *9*, 1735–1780.
10. Shi, X.; Chen, Z.; Wang, H. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Proc. Adv. Neural Inf. Process. Syst.* **2015**, *9199*, 802–810.
11. Masood, S.; Srivastava, A.; Thuwal, H.C. Real-time sign language gesture (word) recognition from video sequences using CNN and RNN. *Intell. Eng. Inf.* **2018**, *695*, 623–632.
12. Lu, N.; Wu, Y.; Feng, L. Deep learning for fall detection: 3D-CNN combined with LSTM on video kinematic data. *IEEE J. Biomed. Health Inf.* **2019**, *23*, 314–323. [CrossRef] [PubMed]
13. Ahmed, A.A.; Aly, S. Appearance-based arabic sign language recognition using hidden Markov models. In Proceedings of the 2014 International Conference on Engineering and Technology (ICET), Cairo, Egypt, 19–20 April 2014; pp. 1–6.
14. Chen, Y.; Luo, B.; Chen, Y.L. A real-time dynamic hand gesture recognition system using kinect sensor. In Proceedings of the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 6–9 December 2015; pp. 2026–2030.
15. Wan, J.; Li, S.Z.; Zhao, Y.; Zhou, S. ChaLearn looking at people RGB-D isolated and continuous datasets for gesture recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 761–769.
16. Xing, Y.; Li, J.; Wang, L. A robust hand gesture recognition method via convolutional neural network. In Proceedings of the 6th International Conference on Digital Home (ICDH), Guangzhou, China, 2–4 December 2016; pp. 64–67.
17. Chai, X.; Liu, Z.; Yin, F. Two streams recurrent neural networks for large-scale continuous gesture recognition. In Proceedings of the 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 15–17 November 2016; pp. 31–36.
18. Naguri, C.R.; Bunescu, R.C. Recognition of dynamic hand gestures from 3D motion data using LSTM and CNN architectures. In Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 1130–1133.
19. Sachara, F.; Kopinski, T.; Gepperth, A. Free-hand gesture recognition with 3D-CNNs for in-car infotainment control in real-time. In Proceedings of the IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–18 October 2017; pp. 959–964.
20. Lin, C.; Wan, J.; Liang, Y.; Li, S.Y. Large-scale isolated gesture recognition using a refined fused model based on masked res-C3D network and skeleton LSTM. In Proceedings of the 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China, 15–19 May 2018; pp. 52–58.

21. Ide, H.; Kurita, T. Improvement of learning for CNN with ReLU activation by sparse regularization. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 2684–2691.
22. Shang, L.; Yang, Q.; Wang, J. Detection of rail surface defects based on CNN image recognition and classification. In Proceedings of the 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon-si Gangwon-do, Korea, 11–14 February 2018; pp. 45–51.
23. Ji, S.; Xu, W.; Yang, M. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 221–231. [CrossRef] [PubMed]
24. Srivastava, N.; Hinton, G.; Krizhevsky, A. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
25. Ronchetti, F.; Quiroga, F.; Estrebou, C. Sign languague recognition without frame-sequencing constraints: A proof of concept on the argentinian sign language. In *Advances in Artificial Intelligence—IBERAMIA 2016*; Springer: Berlin, Germany, 2016; pp. 338–349.
26. Guyon, I.; Athitsos, V.; Jangyodsuk, P. The ChaLearn gesture dataset (CGD 2011). *Mach. Vis. Appl.* **2014**, *25*, 1929–1951. [CrossRef]
27. Azzakhnini, S.; Ballihi, L.; Aboutajdine, D. Learning discriminative features from RGB-D images for gender and ethnicity identification. *J. Electron. Imaging* **2016**, *25*, 061625. [CrossRef]
28. Li, C.; Zhang, X.; Jin, L. LPSNet: A novel log path signature feature based hand gesture recognition framework. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 631–639.

*future internet*

*Article*

# Embedded Deep Learning for Ship Detection and Recognition

**Hongwei Zhao, Weishan Zhang *, Haoyun Sun and Bing Xue**

College of Computer and Communication Engineering, China University of Petroleum (UPC), Qingdao 266580, China; upcvagen@163.com (H.Z.); hys_upc@163.com (H.S.); xueb2016@163.com (B.X.)
* Correspondence: zhangws@upc.edu.cn

Received: 31 December 2018; Accepted: 31 January 2019; Published: 21 February 2019

**Abstract:** Ship detection and recognition are important for smart monitoring of ships in order to manage port resources effectively. However, this is challenging due to complex ship profiles, ship background, object occlusion, variations of weather and light conditions, and other issues. It is also expensive to transmit monitoring video in a whole, especially if the port is not in a rural area. In this paper, we propose an on-site processing approach, which is called Embedded Ship Detection and Recognition using Deep Learning (ESDR-DL). In ESDR-DL, the video stream is processed using embedded devices, and we design a two-stage neural network named DCNet, which is composed of a DNet for ship detection and a CNet for ship recognition, running on embedded devices. We have extensively evaluated ESDR-DL, including performance of accuracy and efficiency. The ESDR-DL is deployed at the Dongying port of China, which has been running for over a year and demonstrates that it can work reliably for practical usage.

**Keywords:** ship identification; fully convolutional network; embedded deep learning; scalability

## 1. Introduction

With the development of the marine economy, marine transportation and management have been attracting more and more attention in modern ports [1]. Ship detection and recognition play an important role for marine transportation management. To accomplish the task of ship detection and recognition, video surveillance with static cameras is a good choice. Surveillance cameras are increasingly deployed for port management and security in order to realize a smart port [2]. However, this is challenging due to complex ship profiles, ship background and object occlusions, variations of weather and light conditions, and other issues.

Deep learning [3] provides a promising technology to tackle these issues. Vehicle plate text recognition is a popular image process method for vehicle identification, which shows promising results for accurate object recognition. The work in [4] handled Chinese car license plate recognition from traffic videos with image features extracted by DCNNs (Deep Convolutional Neural Networks). License plate recognition [5] based on deep learning was also used for feature extraction and classification. This regular character recognition is much simpler than these Chinese characters from ship license plates, due to the usage of various character types and complex backgrounds, and also the variations of ship plate locations.

At the same time, the number of monitor devices can be big, deployed at both the seashore and above the water, which are used to monitor ships sailing in the water and also ships going back and forth from a port. Therefore, the recognition approach requires good scalability and should have the capability to handle a considerate number of video streams. On the other hand, the transmission of all video streams back may not be possible as there may not be Internet connections in some places, and also the cost of using 4G for transmission of video streams is an important factor to design possible recognition solutions.

To address these challenges, we propose an embedded deep learning approach called ESDR-DL (Embedded Ship Detection and Recognition using Deep Learning), in order to conduct ship recognition on the fly by connecting the embedded device to a camera directly. In ESDR-DL, we propose a neural network named DCNet (composed with DNet and Cnet as detailed later) which conducts ship recognition as a classification problem by detecting and identifying key parts of a ship (the bow, the cabin, and the stern), and classifies the ship's identity based on these key parts. These classification results are then voted for the decision of the ship's identity. In order to boost performance, ESDR-DL is designed to handle multi-channel video at the same time. We conduct comprehensive evaluations for the embedded system of ESDR-DL, including the performance of accuracy and efficiency.

The contributions for this paper include:

- To decrease model parameters, we design a tiny network DNet for ship detection, and share the base convolutional layers with CNet.
- To address challenges of variations of ship license plate locations and text types, we propose a classification network CNet to recognize ships.
- We run the DCNet on embedded devices, which has good scalability and can handle a large number of video streams at the same time.

The remainder of the paper is organized as follows: Section 2 discusses the related work. Section 4 presents the architecture design of ESDR-DL. Section 3 discusses the implementation and training steps of DCNet. Section 5 presents comprehensive evaluations of the deployed solution. Section 6 concludes the paper.

## 2. Related Work

We will discuss deep learning and embedded device supported object recognition as ESDR-DL is in principle an embedded deep learning approach, we will also discuss vehicle recognition as ship is also a kind of vehicle.

### *2.1. Deep Learning*

The concept of deep learning originates from the study of artificial neural networks, proposed by Hinton et al. [6]. Deep learning have made remarkable achievements in the field of image processing, especially for object detection. SSD is a typical one stage detector proposed in [7], which processes images in a single network, and and has good efficiency and accuracy. Faster R-CNN [8] is a two-stage detector, which uses RPN (Region Proposal Network) to produce high-quality region proposals and then detect them with Fast R-CNN [9].

Redmon presents a single neural network named YOLO, which abandons anchor boxes, and predicts bounding boxes and class probabilities directly from a full image in one evaluation [10]. YOLO considers object detection as a regression problem to predict bounding boxes and class probabilities. It can be optimized as end-to-end directly with good detection performance. Fast YOLO can process 155 frames per second. Compared with other state-of-the-art detection algorithms, YOLO makes more localization errors.

YOLOV2 [11] is based on YOLO [10]. YOLOV2 removes the fully connected layers from YOLO and uses anchor boxes to predict bounding boxes. The YOLOV2 model can run with various image sizes, and it is easy to make a trade-off between speed and accuracy. YOLOV2 is faster than YOLO, which can process 200 frames per second with the Tiny model. Table 1 shows the performance of these algorithms.

**Table 1.** The performance of the algorithms.

| Algorithm | Datasets | mAP | FPS | Proposed Year |
|---|---|---|---|---|
| SSD300 | VOC 2007 + 2012 | 74.3 | 46 | 2016 |
| SSD500 | VOC 2007 + 2012 | 76.8 | 19 | 2016 |
| Faster-RCNN | VOC 2007 + 2012 | 73.2 | 31 | 2015 |
| YOLO | VOC 2007 + 2012 | 63.4 | 45 | 2015 |
| YOLOv2 | VOC 2007 + 2012 | 76.8 | 67 | 2016 |
| Tiny YOLO | VOC 2007 + 2012 | 57.1 | 207 | 2016 |

### *2.2. Vehicle Identification*

There are many state-of-the-art algorithms that can be used for vehicle detection, such as RCNN [9] YOLO [10], which have high real-time performance, but the accuracy is not high for ship recognition. In [12], Wang proposed a vehicle real-time detection algorithm based on YOLOv2. It optimized important parameters of the model, and improved the number and size of anchors in the model, which can achieve both real-time and high accuracy for vehicle detection. It tested by a home-made dataset, which showed higher accuracy and ran faster than YOLOv2 [11] and RCNN. However, the real-time performance is based on high-performance equipment, which is not suitable for us.

Plate recognition is the most typical application for vehicle identification. Liu et al. [13] proposed CogniMem, which used a neural-network chip to recognize license plates. The CogniMem combined a video image processing module with a neural network module by using an equalized image processing algorithm and network classification algorithm. It contained processes of license location, character segmentation and character recognition. CogniMem can recognize car plates with low error; however, it required that the plates have a fixed character position and limited character type and numbers. Lin [14] proposed a method named ALPR to detect and recognize the characters in the plate region of an image. The approach is not applicable to the situation in which new targets emerge that are not annotated in its database.

### *2.3. Embedded Object Recognition*

Embedded image processing has been attracting a lot of efforts. In [15], Arth et al. designed a full-featured license plate detection and recognition method using DSP. The processing core is a single Texas Instruments fixed point DSP with 1 MB RAM. Additionally, a slower SDRAM memory block of 16 MB exists. It can achieve real-time performance. In addition, Kamat and Ganesan [16] implemented a license plate detection system on a DSP using the Hough transform. Kang et al. [17] implemented a vehicle tracking and license plate recognition system on a PDA. An FPGA was used by Bellas et al. [18] to speed up parts of their license plate recognition system.

There was research that ran the Fast R-CNN on Jetson TK1 platform [19]. Although additional modifications on the Fast R-CNN were made to fit TK1, the detection speed was very low (1.85 frames per second-fps). The work in [20] ran a seven-layer CNN on TDA3x SoC for object classification, and the overall system performance was 15 fps. Therefore, a powerful software/hardware platform is needed to support efficient embedded deep learning based real-time video processing.

## **3. Designing a Recognition Neural Network-DCNet**

DCNet is a two-stage network that consists of a DNet and a CNet as shown in Figure 1. DNet is a fully convolutional network [21] for ship parts detection including ship bow, cabin and stern. CNet is a classifier that can takes an image of any size and output a set of classification scores. We locate the ship parts from the DNet, and feed them into the CNet to get three classification scores (bow score, cabin score, stern score) of ship identify. Finally, a voter is used to recognize the ship as shown in Figure 1.

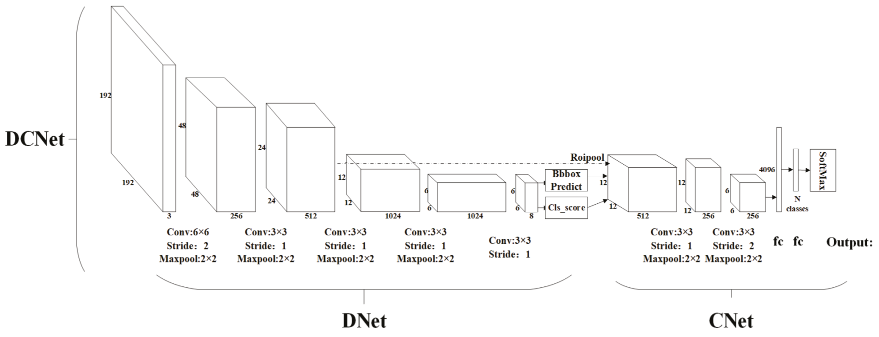

DCNet
192
192
3
48
48
256
24
24
512
12
12
1024
6
6
1024
6
6
8
Bbbox Predict
Cls_score
Roipool
12
12
512
12
12
256
6
6
256
4096
N classes
SoftMax
Conv:6×6 Stride: 2 Maxpool:2×2
Conv:3×3 Stride: 1 Maxpool:2×2
Conv:3×3 Stride: 1 Maxpool:2×2
Conv:3×3 Stride: 1 Maxpool:2×2
Conv:3×3 Stride: 1
Conv:3×3 Stride: 1 Maxpool:2×2
Conv:3×3 Stride: 2 Maxpool:2×2
fc
fc
Output:
DNet
CNet

**Figure 1.** DCNet.

## *3.1. DNet*

Region Proposal is one of the key points for a target detection network, such as Faster R-CNN [8] using RPN to generate better regions, and YOLOv1 [10] splitting an image into grid cells as region proposals to improve the detection efficiency. As shown in Figure 2, DNet divides the input image into 6 $\times$ 6 grid cells as region proposals like YOLOv1 [10]. Each region proposal consists of eight predictions: $x, y, w, h, c, C \times 3$. The $(x, y)$ coordinates represent the center of the predicted box. $w$ and $h$ represent the width and height of the predicted box. $c$ represents the IOU (intersection-over-union) between the predicted box and ground truth box. $C \times 3$ represents the probability classes of bow, cabin and stern.

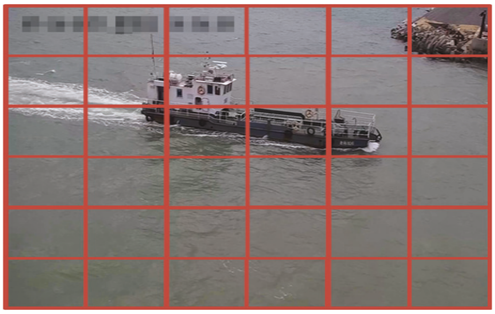


**Figure 2.** Region proposal of DNet.

Inspired by YOLOv1, DNet models the detection as a regression problem. Since the object and background are relatively simple, object features are relatively obvious, and the network is not as deep as VGGNet [10] and ResNet [22]. We pay more attention to the decreasing network model parameters. DNet resizes the image to 192 $\times$ 192 as the input and we design five layers to extract features from an image; the last two layers predict the object probabilities and its coordinates.

DNet predicts bounding boxes based on grid cells. A grid cell produces one bounding box predictor. We need one bounding box predictor to be "responsible" for each object, and choose the one based on which prediction has the highest current IOU (intersection-over-union) with the ground truth. To choose a proper predictor for each object, at training time, we design the loss $L_g$ function as follows:

$$L_{grid} = \sum_{i=0}^{S^2} (C_i - C_i^*)^2, \tag{1}$$

where $S^2$ is the number of the grid cells, $C_i$ is the confidence value that the predicting box contain an object, and $C_i^*$ is the IOU between predicted bounding box with ground truth. If there is no object in predictions, then $C_i^* = 0$.

The final layer predicts both class probabilities and bounding box coordinates; we calculate the coordinates loss and classification loss only when the predictor is a proper one, the loss function is:

$$L_{box} = \sum_{c \in propers} \sqrt{(x_c - x_c^*)^2 + (y_c - y_c^*)^2 + (w_c - w_c^*)^2 + (h_c - h_c^*)^2} + (p_c - p_c^*)^2, \tag{2}$$

where $p_c$ is the predicting class and $p_c^*$ is the truth class. The loss of $L_{box}$ is under the assumption that the predictor is a proper one. Therefore, it may not be ideal to weight the $L_{grid}$ equally to $L_{box}$. We use $\lambda$ to weight the loss, and the final loss function is designed as follows:

$$L = \lambda L_{grid} + (1 - \lambda) L_{box}. \tag{3}$$

### 3.2. CNet

Ship recognition is challenging, and we can make use of the fact that there is only a limited number of ships in a port. CNet model considers the recognition problem as a classification problem, which is connected to the end of the DNet. We set the output boxes and classes as the input and share the first three layers' feature maps of the DNet. The boxes is resized to 14 × 14 by a ROIPool layer as shown in Figure 3, which was proposed in [8]. Two extra convolutional layers followed by the ROIPool layer are added, and, finally, two fully connected layers and a softmax layer are used to predict the output probabilities.

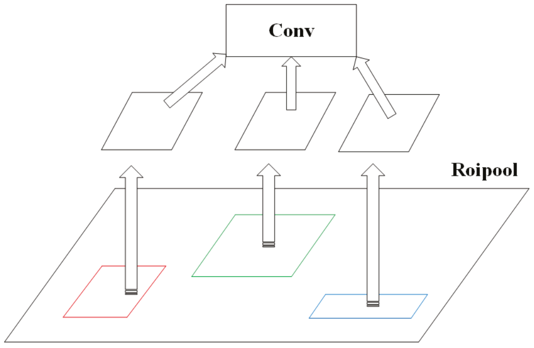


**Figure 3.** ROIPool Layer.

Finally, CNet outputs three ship classification scores of bow, cabin and stern. We design the voting strategy as

$$S_{i:i \in (probabilities)} = \lambda_b Score^i_{bow} + \lambda_c Score^i_{cabin} + \lambda_s Score^i_{stern}.$$

$Score^i$ denotes the output score of probabilities $i$. It weights the score of cabins equally with scores of bow and stern which may not be ideal. To resolve this, we use $\lambda$ to weight the scores.

### 3.3. Training and Running

Before training, we have to label the ship data set. We quadrangle the bow, cabin and stern with ($c$,$i$,$x$,$y$,$w$,$h$). $c$ represents the key point of ships, and $i$ represents the identification of the key point, ($x$,$y$) represents the upper-left coordinates of the box, $w$ represents the with of the box and $h$ represent the hight of the box. To learn the shared features, we train the DCNet with two steps as shown in Figure 4. In the first step, we train the DCNet use the ship data set, we set the initial learning rates as 0.01 and decrease by one tenth per 10,000 iterations; after 50,000 iterations, the losses tend to stabilize.

In the second step, we fix the shared convolutional layers and only fine-tune the unique layers of CNet. During CNet training, we feed the ship data sets to the shared convolutional layers and rectangle the box feature maps, unify the box feature maps size by the ROIPool layer, and, lastly, classify the feature maps with the unique layers of CNet. We set the initial learning rates as 0.1 and decrease by one-tenth per 5000 iterations; after 40,000 iterations, the loss tends to be stabilized.

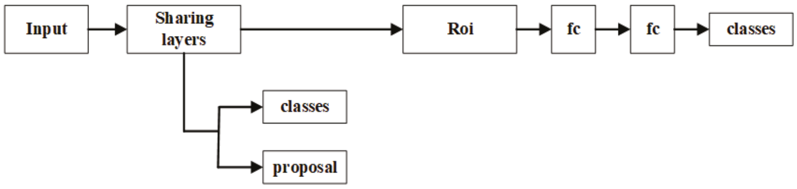


**Figure 4.** DCNet Training.

When running the model, firstly, the DNet predicts the coordinates and classes of bow, cabin and stern, and then it rectangles key ship parts from the sharing feature maps and feeds them to the CNet to get the probability scores, as shown in Figure 5.

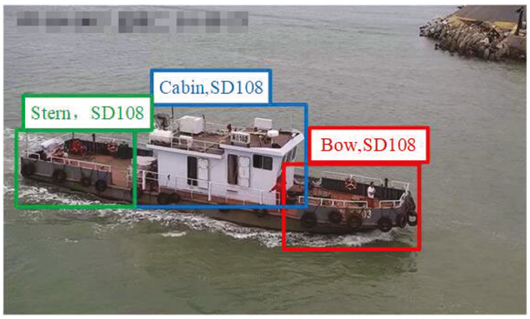


**Figure 5.** Labels of bow, cabin and stern.

## 4. Architecture Design of ESDR-DL

In order to reduce network traffic caused by video streaming from surveillance cameras, and resolve the limitation of low transmission bandwidth, we design an embedded architecture for deep learning, which connects surveillance cameras and performs image processing at the front end, as shown in Figure 6. In this ESDR-DL, the video stream is connected to a nearby TX2 through a LAN. To ensure real-time performance of video surveillance, each TX2 receives only one or two video streams. When the system is running, a Video Stream Receiver in TX2 is responsible for receiving the video stream accessed by the current device, decoding the video stream through a Video Stream Decoder, and inputting the decoded images to an Image Processor for detection. In the Image Processor, the DCNet model is used to detect and identify key parts of a ship (the bow, the cabin, and the stern), and classify the ship's identity based on these key parts, and output three prediction results. These prediction results are then used in a Voter for the decision of the ship's identity.

We use NVIDIA Jetson TX2 as it is an industry-leading embedded computing device. Table 2 lists the main properties of TX2 related to our work in this paper.

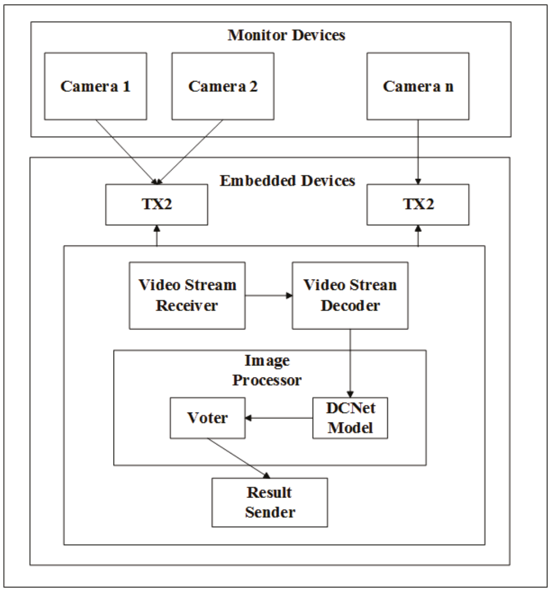


**Figure 6.** System architecture of ESDR-DL.

**Table 2.** Jetson TX2.

| Versions | Jeston TX2 |
|---|---|
| GPU | NVIDIA Pascal 256 CUDA core |
| CPU | 64-bit Denver 2 and A57 CPUs |
| Memory | 8 GB 128-bit LPDDR4 |
| Storage | 32 GB eMMC |
| Video Encode | 4 K × 2 K 60 Hz |
| Video Decode | 4 K × 2 K 60 Hz |
| Camera | 1.4 Gpix/s 2.5 Gbps per lane |
| Connectivity | 1 Gigabit Ethernet, 802.11ac WLAN |

## 5. Experiment Results

We use the recall-R and precision-P as the evaluation standard, defined as

$$R = TP/(TP + FN)\,,\ P = TP/(TP + FP).$$

TP refers to true positive, FN indicates false negative, and FP means false positive.

### 5.1. Algorithm Performance

To evaluate the performance of DNet, we use a ship data set that has 6000 images collected from Donging port, Shandong, China. We have tested both Yolo Tiny and DNet, running on TX2 and GTX TITAN X. In addition, 4700 images are used for training and 1300 are used for testing. Table 3 shows the test results. We can see that DNet achieves much higher energy efficiency with a little lower accuracy.

**Table 3.** Test results of Tiny YOLO and DNet.

| Method | Device | Accuracy | Efficiency (FPS) | Power (w) | Energy Efficiency (fps/w) |
|---|---|---|---|---|---|
| Tiny YOLO | TX2 | 0.9316 | 18.91 | 7.84 | 2.41 |
| DNet | TX2 | 0.9233 | 34.87 | 7.73 | 4.51 |
| Tiny YOLO | GTX TITAN X | 0.9492 | 155.24 | 180 | 0.86 |
| DNet | GTX TITAN X | 0.9297 | 298.43 | 177 | 1.68 |

YOLOv1 splits an image into $7 \times 7$ grid cells; considering the big target of ship and the limitation of computing capacity of a TX2, we decrease the grid cells to reduce model parameters. As is shown in Table 4 where efficiency is measured by FPS (Frames Per Seconds), a test is made to check this, and DNet splits an image into $6 \times 6$, considering the performance–accuracy trade off.

**Table 4.** Grid cells number test for DNet.

| Grid Cells Number | Detection Accuracy | Efficiency |
|---|---|---|
| $4 \times 4$ | 0.7642 | 59 |
| $5 \times 5$ | 0.8310 | 51 |
| $6 \times 6$ | 0.9233 | 43 |
| $7 \times 7$ | 0.9251 | 38 |
| $8 \times 8$ | 0.9282 | 32 |
| $9 \times 9$ | 0.9263 | 21 |

The $\lambda$ for loss L in Equation (3) can be changed based on different scenarios and targets. We adjust $\lambda$ experimentally and the results is shown in Table 5. Concluding from the tests, we set $\lambda = 0.7$.

**Table 5.** $\lambda$ test for Loss L.

| $\lambda$ | Detection Accuracy |
|---|---|
| 0.1 | 0.5216 |
| 0.2 | 0.7442 |
| 0.3 | 0.7513 |
| 0.4 | 0.8121 |
| 0.5 | 0.9021 |
| 0.6 | 0.9113 |
| 0.7 | 0.9233 |
| 0.8 | 0.9035 |
| 0.9 | 0.8945 |

We adjust $\lambda$ experimentally for voting strategy and test its impact on accuracy as in Table 6. We can conclude the weights from it that $\lambda_c > \lambda_b > \lambda_s$. We set $\lambda_b = 0.3$, $\lambda_c = 0.5$ and $\lambda_s = 0.2$.

**Table 6.** $\lambda$ impacts on recognition accuracy.

| $\lambda_b$ | $\lambda_c$ | $\lambda_s$ | Accuracy |
|---|---|---|---|
| 1 | 0 | 0 | 0.83 |
| 0 | 1 | 0 | 0.85 |
| 0 | 0 | 1 | 0.80 |
| 0.33 | 0.34 | 0.33 | 0.84 |
| 0.25 | 0.5 | 0.25 | 0.85 |
| 0.3 | 0.5 | 0.2 | 0.86 |
| 0.2 | 0.5 | 0.3 | 0.85 |

## 5.2. System Performance

ESDR-DL is deployed to Dongying port, China. The video cameras used are Hikvision DS-2CD3T25D-I5. The pixel used is 1920 × 1080 and the frame rate is 30 fps. We use seven TX2s for 10 cameras as shown in Table 7. Four cameras are installed on both sides of the entrance with a height of 8 m. Others are installed inside the port.

**Table 7.** Deployment of cameras and TX2s.

| | |
|---|---|
| TX2-1 | entrance camera-1 |
| TX2-2 | entrance camera-2 |
| TX2-3 | entrance camera-3 |
| TX2-3 | entrance camera-4 |
| TX2-4 | inside port camera-1 |
| TX2-5 | inside port camera-2 |
| TX2-6 | inside port camera-3 |
| TX2-6 | inside port camera-4 |
| TX2-7 | inside port camera-5 |
| TX2-7 | inside port camera-6 |

During one month's running, we collect 13,000 recognized records and checks the accuracy manually. There are a total of 14,536 ships in videos.

Table 8 illustrates the recall and precision rates of ship detection and recognition. S denotes the ship number occurring in each camera, D-P stands for the ship detection precision, D-R is the detection recall, R-P is the recognition precision, R-R refers to the recognition recall, and T denotes the processing efficiency of each camera.

**Table 8.** Performance of ESDR-DL.

| Camera | S | D-P | D-R | R-P | R-R | T |
|---|---|---|---|---|---|---|
| entrance camera-1 | 903 | 0.86 | 0.80 | 0.82 | 0.74 | 27 fps |
| entrance camera-2 | 903 | 0.86 | 0.79 | 0.82 | 0.74 | 27 fps |
| entrance camera-3 | 887 | 0.85 | 0.79 | 0.81 | 0.74 | 27 fps |
| entrance camera-4 | 891 | 0.86 | 0.80 | 0.80 | 0.73 | 27 fps |
| inside port camera-1 | 1532 | 0.89 | 0.84 | 0.84 | 0.79 | 13 fps |
| inside port camera-2 | 1129 | 0.87 | 0.82 | 0.80 | 0.75 | 13 fps |
| inside port camera-3 | 1413 | 0.90 | 0.85 | 0.85 | 0.78 | 13 fps |
| inside port camera-4 | 1410 | 0.89 | 0.84 | 0.84 | 0.79 | 13 fps |
| inside port camera-5 | 1611 | 0.89 | 0.85 | 0.82 | 0.79 | 13 fps |
| inside port camera-6 | 1611 | 0.88 | 0.85 | 0.82 | 0.80 | 13 fps |

Comparing Tables 6 and 8, we can find that the accuracy of actual running is lower than the accuracy testing in the home-made data set because there are new ships arriving the port and the ESDR-DL can not recognize these new ships. In addition, ESDR-DL performs better for the inside-port monitoring cameras because there are some far away scenes of ships from the entrance cameras and only close scenes exist from the inside-port cameras, while DCNet focus on big target detection and recognition. In addition, as shown in Figure 7, the system can run in bad weather conditions (such as rain and smog) in practice. In order to test the performance of the system in bad weather, we run the system in rain and smog weather, and run it at dusk (5:00 p.m.–6:00 p.m.). The detection recognition results are shown in Table 9.

The recognition results are shown in Table 10. We can see that the accuracy of the system is dropping sharply in rain and smoggy weather, while performing well at dusk. This is not a problem in practice as there are very few ships in such weather conditions.

**Table 9.** Performance of ship detection in bad weather.

| Camera | Rain-S | Rain-P | Ran-R | Smog-S | Smog-P | Smog-R | Dusk-S | Dusk-P | Dusk-R |
|---|---|---|---|---|---|---|---|---|---|
| entrance camera-1 | 46 | 0.72 | 0.65 | 114 | 0.61 | 0.52 | 203 | 0.83 | 078 |
| entrance camera-2 | 46 | 0.74 | 0.65 | 114 | 0.59 | 0.51 | 203 | 0.87 | 0.80 |
| entrance camera-3 | 39 | 0.69 | 0.56 | 99 | 0.49 | 0.39 | 211 | 0.85 | 0.79 |
| entrance camera-4 | 40 | 0.70 | 0.55 | 103 | 0.48 | 0.40 | 225 | 0.83 | 0.80 |
| inside port camera-1 | 70 | 0.75 | 0.70 | 91 | 0.64 | 0.51 | 293 | 0.82 | 0.76 |
| inside port camera-2 | 58 | 0.71 | 0.64 | 69 | 0.60 | 0.55 | 233 | 0.80 | 0.71 |
| inside port camera-3 | 132 | 0.79 | 0.71 | 155 | 0.70 | 0.53 | 254 | 0.83 | 0.79 |
| inside port camera-4 | 140 | 0.74 | 0.70 | 169 | 0.67 | 0.59 | 254 | 0.85 | 0.80 |
| inside port camera-5 | 129 | 0.78 | 0.73 | 143 | 0.72 | 0.56 | 223 | 0.85 | 0.76 |
| inside port camera-6 | 129 | 0.75 | 0.71 | 143 | 0.70 | 0.51 | 223 | 0.80 | 0.77 |

**Table 10.** Performance of ship recognition in bad weather.

| Camera | Rain-S | Rain-P | Ran-R | Smog-S | Smog-P | Smog-R | Dusk-S | Dusk-P | Dusk-R |
|---|---|---|---|---|---|---|---|---|---|
| entrance camera-1 | 46 | 0.53 | 0.45 | 114 | 0.36 | 0.20 | 203 | 0.79 | 0.72 |
| entrance camera-2 | 46 | 0.45 | 0.38 | 114 | 0.31 | 0.18 | 203 | 0.81 | 0.75 |
| entrance camera-3 | 39 | 0.46 | 0.36 | 99 | 0.29 | 0.21 | 211 | 0.82 | 0.74 |
| entrance camera-4 | 40 | 0.45 | 0.32 | 103 | 0.26 | 0.15 | 225 | 0.83 | 0.73 |
| inside port camera-1 | 70 | 0.55 | 0.41 | 91 | 0.34 | 0.21 | 293 | 0.80 | 0.72 |
| inside port camera-2 | 58 | 0.52 | 0.45 | 69 | 0.29 | 0.17 | 233 | 0.75 | 0.70 |
| inside port camera-3 | 132 | 0.49 | 0.39 | 155 | 0.39 | 0.23 | 254 | 0.80 | 0.74 |
| inside port camera-4 | 140 | 0.54 | 0.45 | 169 | 0.35 | 0.19 | 254 | 0.82 | 0.76 |
| inside port camera-5 | 129 | 0.48 | 0.43 | 143 | 0.32 | 0.24 | 223 | 0.80 | 0.75 |
| inside port camera-6 | 129 | 0.51 | 0.41 | 143 | 0.25 | 0.12 | 223 | 0.84 | 0.75 |

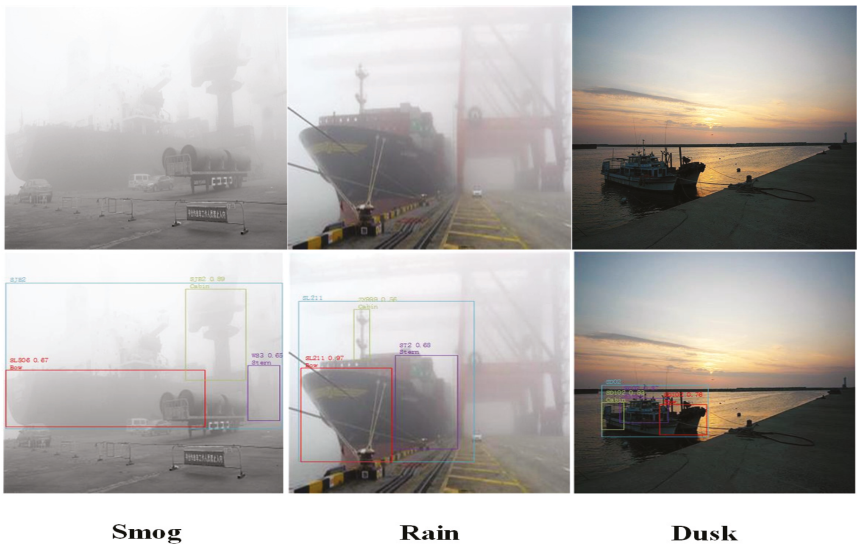


**Figure 7.** Ships in bad weather: The top is the ships in bad weather, and the bottom is the process results.

## 6. Conclusions

Considering the challenges of ship detection and recognition, this paper proposes an embedded deep learning system for ship detection and recognition named ESDR-DL. It first locates the bow, cabin and stern of the ship using DNet, and then recognizes them by a classification network named CNet. Finally, voting is used to recognize the ship identification. We implement the ESDR-DL with an embedded architecture which supports real-time video processing. We have deployed ESDR-DL at

Dongying port, China. It has been running stably in the past year, which shows the effectiveness of our solution. In the future, we will adopt a multi-model data fusion approach [23,24] to improve the recognition accuracy.

**Author Contributions:** Conceptualization, H.Z. and W.Z.; methodology, H.Z.; validation, H.Z., H.S. and B.X.; formal analysis, W.S.; investigation, B.X.; resources, W.Z.; writing–original draft preparation, H.Z.; writing–review and editing, H.Z. and H.S.; visualization, W.Z. and B.X.; project administration, W.Z.

**Funding:** This research was funded by the Key Research Program of Shandong Province under Grant No. 2017GGX10140 and the National Natural Science Foundation of China under Grant No. 61309024.

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

## Nomenclature

The following abbreviations are used in this manuscript:

| | |
|---|---|
| ESDR-DL | Embedded Ship Detection and Recognition using Deep Learning |
| DCNet | Detection and Classification Network |
| DNet | Detection Network |
| CNet | Classification Network |
| DCNN | Deep Convolutional Neural Network |
| RPN | Region Proposal Network |
| RCNN | Region-based Convolutional Neural Network |
| ROI | Region Of Interest |
| DSP | Digital Signal Processing |
| PDA | Personal Digital Assistant |
| FPGA | Field-Programmable Gate Array |
| TDA3x SoC | Threat Discovery Appliance |
| SSD | Single Shot MultiBox Detector |
| YOLO | You Only Look Once |

## References

1. Wang, Z.; Tang, W.; Zhao, L. Research on the modern port logistics development in the city-group, China. In Proceedings of the 2010 International Conference on IEEE Logistics Systems and Intelligent Management (ICLSIM), Harbin, China, 9–10 January 2010; pp. 1280–1283.
2. Alderton, P.M. *Port Management and Operations*; Harbors: Suffolk, NY, USA, 2008.
3. Xu, L.; Ren, J.S.J.; Liu, C.; Jia, J. Deep convolutional neural network for image deconvolution. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 1790–1798.
4. Zang, D.; Chai, Z.; Zhang, J.; Zhang, D.; Cheng, J. Vehicle license plate recognition using visual attention model and deep learning. *J. Electron. Imaging* **2015**, *24*, 033001. [CrossRef]
5. Masood, S.Z.; Shu, G.; Dehghan, A.; Ortiz, E.G. License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks. *arXiv* **2017**, arXiv:1703.07330.
6. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
7. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
8. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*. [CrossRef] [PubMed]
9. Girshick, R. Fast r-cnn. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015. [CrossRef]
10. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2015**, arXiv:1506.02640v5,

11. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *arXiv* **2016**, arXiv:1612.08242.
12. Wang, H.; Zhang, Z. A vehicle real-time detection algorithm based on YOLOv2 framework. In Proceedings of the Real-Time Image and Video Processing, Orlando, FL, USA, 15–19 April 2018; p. 22.
13. Liu, Y.; Wei, D.; Zhang, N.; Zhao, M. Vehicle-license-plate recognition based on neural networks. In Proceedings of the 2011 IEEE International Conference on Information and Automation, Shenzhen, China, 6–8 June 2011; pp. 363–366.
14. Lin, D.; Lin, F.; Lv, Y.; Cai, F.; Cao, D. Chinese Character CAPTCHA Recognition and Performance Estimation via Deep Neural Network. *Neurocomputing* **2018**, *28*, 11–19. [CrossRef]
15. Arth, C.; Limberger, F.; Bischof, H. Real-Time License Plate Recognition on an Embedded DSP-Platform. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition 2007, Minneapolis, MN, USA, 17–22 June 2007. [CrossRef]
16. Kamat, V.; Ganesan, S. An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S. In Proceedings of the Real-Time Technology and Applications Symposium, Chicago, IL, USA, 15–17 May 1995; pp. 58–59.
17. Kang, J.S.; Kang, M.H.; Park, C.H.; Kim, J.H.; Choi, Y.S. Implementation of embedded system for vehicle tracking and license plates recognition using spatial relative distance. In Proceedings of the International Conference on Information Technology Interfaces, Cavtat, Croatia, 7–10 June 2003; Volume 1, pp. 167–172.
18. Bellas, N.; Chai, S.M.; Dwyer, M.; Linzmeier, D. FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators. In Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes Island, Greece, 25–29 April 2006.
19. Mao, H.; Yao, S.; Tang, T.; Li, B.; Yao, J.; Wang, Y. Towards real-time object detection on embedded systems. *IEEE Trans. Emerg. Top. Comput.* **2016**, *6*, 417–431. [CrossRef]
20. Jagannathan, S.; Desappan, K.; Swami, P.; Mathew, M.; Nagori, S.; Chitnis, K.; Marathe, Y.; Poddar, D.; Narayanan, S. Efficient object detection and classification on low power embedded systems. In Proceedings of the 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 8–10 January 2017; pp. 233–234.
21. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440, doi:10.1109/CVPR.2015.7298965. [CrossRef]
22. 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.
23. Zhang, W.; Zhang, Y.; Zhai, J.; Zhao, D.; Xu, L.; Zhou, J.; Li, Z.; Yang, S. Multi-source data fusion using deep learning for smart refrigerators. *Comput. Ind.* **2018**, *95*, 15–21. [CrossRef]
24. Zhang, W.; Wang, Z.; Liu, X.; Gong, W.; Sun, H.; Zhou, J.; Liu, Y. Deep Learning based Real-Time Fine-grained Pedestrian Recognition using Stream Processing. *IET Intell. Transp. Syst.* **2018**, *12*. [CrossRef]

*future internet*

Article

# Tooth-Marked Tongue Recognition Using Gradient-Weighted Class Activation Maps

**Yue Sun [1], Songmin Dai [1], Jide Li [1], Yin Zhang [1] and Xiaoqiang Li [1,2,*]**

1 School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; flyyue@shu.edu.cn (Y.S.); laodar@shu.edu.cn (S.D.); iavtvai@shu.edu.cn (J.L.); zhangyin6998@t.shu.edu.cn (Y.Z.)

2 Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China

* Correspondence: xqli@shu.edu.cn

Received: 11 January 2019; Accepted: 13 February 2019; Published: 15 February 2019

**Abstract:** The tooth-marked tongue is an important indicator in traditional Chinese medicinal diagnosis. However, the clinical competence of tongue diagnosis is determined by the experience and knowledge of the practitioners. Due to the characteristics of different tongues, having many variations such as different colors and shapes, tooth-marked tongue recognition is challenging. Most existing methods focus on partial concave features and use specific threshold values to classify the tooth-marked tongue. They lose the overall tongue information and lack the ability to be generalized and interpretable. In this paper, we try to solve these problems by proposing a visual explanation method which takes the entire tongue image as an input and uses a convolutional neural network to extract features (instead of setting a fixed threshold artificially) then classifies the tongue and produces a coarse localization map highlighting tooth-marked regions using Gradient-weighted Class Activation Mapping. Experimental results demonstrate the effectiveness of the proposed method.

**Keywords:** tooth-marked tongue; convolutional neural network; gradient-weighted class activation maps

## 1. Introduction

Inspection of the tongue is one of the most important diagnostic methods in traditional Chinese medicine (TCM). According to [1], medical experts diagnose diseases by observing patient tongue color, tongue shape, and other characteristics of the tongue. Different features of the tongue reflect the internal state of the body and the health of the organs. Thus, tongue diagnosis has been widely applied to clinical analysis for thousands of years [2]. Tooth-marked tongue, a kind of abnormal tongue, is one appearance of the tongue when there are teeth marks along the lateral borders [3]. Medical experts believe that the tooth-marked tongue is caused by spleen deficiency, which provides guidance for clinical syndrome differentiation [4]. The appearances of the tooth-marked tongue are shown in Figure 1; (a) is a normal tongue image for reference. (b.1) and (b.2) are tooth-marked tongue images with teeth-marked regions shown in blue boxes. According to previous surveys, the incidence of tooth-marked tongue in the crowd is about 56%, of which the severe ones accounts for 11% [5]. However, the recognition of tooth-marked tongue is a challenging task for TCM practitioners. The appearance of tooth-marked tongues has a great number of variations, such as different colors, different shapes, and different types of teeth marks [6]. Therefore, clinical effectiveness of the diagnosis heavily depends on the TCM practitioner's experience. For this reason, more and more computer researchers have begun to combine image processing with pattern recognition technology to establish an objective and quantitative TCM recognition system [7,8].

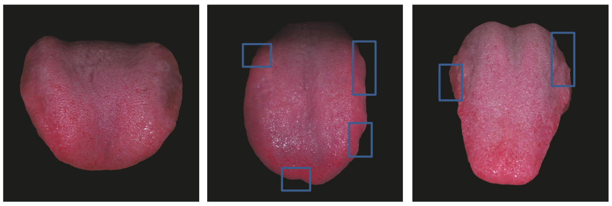


**Figure 1.** Examples of a heathy tongue and tooth-marked tongues. The tooth-marked regions in (b.1) are obvious, while the tooth-marked regions on (b.2) are difficult to identify.

The recognition of tooth-marked tongues can be viewed as a fine-grained classification problem, but it is more challenging than distinguishing between subcategories due to some specific difficulties in the field of tongue diagnosis. Firstly, the number of tongue images is limited because of personal privacy and image acquisition limitations. Secondly, a tongue image is labeled as a tooth-marked tongue or a nontooth-marked tongue, and the locations of the tooth-marked regions are not available. Moreover, existing approaches have a lack of decomposability into intuitive and understandable components, making tongue diagnoses hard to interpret. These questions lead us to seek help from Gradient-weight Class Activation Mapping (Grad-CAM). Grad-CAM was proposed by Selvaraju et al. [9] to provide visual explanations of the Convolutional Neural Network (CNN). It uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept. It was shown that even if there is no location information when training a classification network, convolutional neural networks still have remarkable abilities for localizing objects. In this work, we adopted the Grad-CAM technique to help us analyze the tooth-marked tongue. We present a method that accurately classifies tooth-marked tongue and localizes the important regions in the image for predicting the pathology without bounding boxes. Through the visual interpretation of the tooth-mark problem, we also explore the effect of different receptive field sizes on the classification results. The experimental result shows that our method provides excellent interpretability while improving tongue recognition accuracy.

The remainder of this paper is organized as follows. Section 2 reviews the related work briefly. Section 3 describes the proposed method for tooth-marked tongue recognition in detail. Section 4 presents the detailed process of the method and results of experiments. Finally, this study is concluded in Section 5.

## 2. Related Work

### *2.1. Tongue Diagnosis*

In the past few decades, some researchers have been contributing to the field of computerized tongue diagnosis, including tongue examination system establishment and tongue analysis. Chiu et al. [10] built a computerized tongue examination system for the purpose of quantizing the tongue properties in traditional Chinese medical diagnoses. Zhang et al. [11] established the relationship between tongue appearances and diseases using Bayesian network classifiers based on quantitative features. Many works have also proposed techniques for tongue segmentation [12], tongue image color analysis [13,14], and tongue shape analysis [15].

In the study of the tooth-marked tongue, the threshold of tongue concavity is an important indicator for classifying the tooth-marked tongue. Zhang [16] pointed out that the tooth-marked tongue is very common in tongue images. It is fatter than the normal tongue, the texture is more tender, and the color is paler. Li [17] proposed a method, based on specific thresholds, to extract features of tooth-marked tongues. Firstly, in order to find suspicious tooth-marked regions, he set a threshold for the curvature change of the tongue edge. Secondly, he scanned the edge of the tongue image with a diamond-shaped box. Finally, the R-value of the box, which represents the color of the tongue image, was defined as a feature to classify the tooth-marked tongue. Wang et al. [18] calculated the slope and length of the tongue image, and used the threshold of this information to identify tooth-marked tongues. Shao et al. [19] defined features of tongues which focused on the change of curvature and brightness. They classified tooth-marked tongues by thresholding these feature values. Recently, some researchers have used CNN features to extract tooth-marked features. In [6], a method for extracting features using CNN, using a multi-instance classifier for final classification, was proposed.

### 2.2. Visual Explanation

CNN have significantly improved the performance of many computer vision tasks, such as image classification [20] and object detection [21]. There have been many recent studies exploring CNN visualizations. Zeiler et al. [22] used deconvolutional networks to visualize what patterns activate each unit and discovered the performance contribution from different model layers. Springenberg et al. [23] used guided backpropagation to make modifications to 'raw' gradients that resulted in qualitative improvements. Zhou et al. [24] indicated that a CNN learns object detectors while being trained to identify scenes. They proved that in a single forward-pass, the same network can perform both object classification and object localization. Mahendran et al. [25] reversed the characteristics of different convolutional layers and analyzed the visual coding of CNN. They showed that certain layers in the CNN retain accurate information, such as varying degrees of geometric features. Class Activation Mapping (CAM) was proposed by Zhou et al. [24]. This approach highlighted the class-specific discriminative regions by modifying the image classification CNN architecture. It replaced fully-connected layers with convolutional layers and global average pooling, and generated the CAM by mapping back the predicted category score to the previous convolutional layer. These methods are not only suitable for common datasets, but also for a variety of medical imaging tasks, such as cancer classification [26] and pneumonia detection [27].

## 3. Method

### 3.1. Problem Formulation

The tooth-marked tongue recognition task is a binary classification problem, where the input is a picture $X$ taken in the standard image acquisition environment [14], and the output is a binary label $y \in \{0, 1\}$ indicating the absence or presence of tooth-marked tongue, respectively. For each example in the training set, we optimize the weighted binary cross entropy loss

$$L(X, y) = -w_+ \cdot y logp(Y = 1|X) - w_- \cdot (1 - y) logp(Y = 0|X), \tag{1}$$

where $p(Y = i|X)$ is the probability that the network assigns to the label $i$, $w_+ = |N|/(|P| + |N|)$, and $w_- = |P|/(|P| + |N|)$ with $|P|$ and $|N|$ the number of positive samples and negative samples of tooth-marked tongue in the training set, respectively.

### 3.2. Model Architecture

As stated in Section 1, robust features, which can combine color, shape, and texture information of tongues, are needed to describe the tooth-marked symptom. In this paper, we use the CNN to extract a fixed-length feature vector of the tongue image. As shown in Figure 2, the proposed method takes a tongue image as input and outputs the probability of tooth-marked along with a heatmap localizing the most indicative tooth-marked regions in the image.

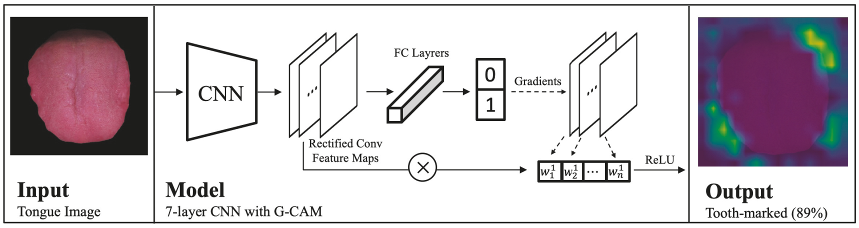


**Figure 2.** Our method is designed to output the probability of a tooth-marked tongue and localize regions in the image most indicative of the pathology. In this example, given a tongue image as input, we forward-propagate the image through the Convolutional Neural Network (CNN) and then compute a raw score (89%) for the class of tooth-marked tongue. Then, we reset the output to one for tooth-marked tongue predictions while zero for nontooth-marked ones. This signal is further backpropagated to the rectified convolutional feature maps of interest to acquire their corresponding gradients, which we combine to compute the coarse Gradient-weight Class Activation Mapping (Grad-CAM) localization (heatmap) which represents where the model has to look to make the particular decision.

The proposed network has seven weight layers, five of which are convolutional layers and the rest of which are fully connected layers (FC layers). Input images are downscaled to 256 × 256 and randomly cropped to 224 × 224. The convolution kernel has a size of 3 × 3 and a stride of 1, and the kernel channel for each convolutional layer is 128, except that the first layer is 64. We apply max-pooling with a size of 2 × 2 with a stride of 2 on each feature map to reduce the filter responses to a lower dimension. Instead of traditional sigmoid or tanh neurons, we use Rectified Linear Units (ReLUs) in each convolution layer and the full connection layer [20], which enables the network to converge several times faster while achieving almost identical performance. We use 0.7 dropout, followed by the fifth pooling layer, to reduce overfitting in the model training procedures. The last FC layer is a 2-way fully connected layer, that represents whether the image is a tooth-mark tongue or not. We use softmax to output the probability of each category as a classification function.

Many previous works have shown that the fully-connected layers lose spatial information about the image, but the convolutional layers naturally preserve this information, while deeper features can capture higher levels of the visual construct. Therefore, in [28], it is conjectured that the last convolutional layers have the best expression between abstract semantics and specific spatial information, and the neurons in these convolutional layers can look up the semantic information of a particular class. Grad-CAM uses the gradient values of different convolutional layers to analyze the importance of each neuron for classification [9]. In order to generate the class-discriminative localization map, $L_{Grad-CAM} \in \mathbb{R}^{u \times v}$ of width $u$ and height $v$ for tooth-marked tongue class, the score of the gradient for tooth-marked tongue class $y$ is calculated, and the feature maps $A^k$ to a convolutional layer is obtained (i.e., $\frac{\partial y}{\partial A^k}$). These gradients are fed back into global average pooling to obtain the weights of the neurons for the tooth-marked tongue $\alpha_k$:

$$\alpha_k = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y}{\partial A^k_{ij}}}_{\text{gradients via backprop}} . \tag{2}$$

This weight $\alpha_k$ represents the weight of feature map $k$ to the tooth-marked tongue class, and $\partial A^k_{ij}$ represents the pixel value at the $(i, j)$ position in the feature map $k$. Global Average Pooling (GAP) [24] outputs the spatial average of each unit in the feature map. After obtaining the weights of the tooth-marked tongue class for all feature maps, the weights can be summed to obtain the heat map.

We calculate the weighted combination of forwarding activation maps, and further process the results by a ReLU function,

$$L_{Grad-CAM} = ReLU(\sum_k \alpha_k A^k). \tag{3}$$

We apply the ReLU function to the linear combination of maps because we only focus on the features that have a positive impact on the tooth-marked tongue. In the heat map, the highlighted areas represent pixels that contribute a large amount to the tooth-marked tongue classification.

## 4. Experiment and Discussion

In this section, we present four different experiments results of the proposed method. The first is the result on five-fold cross-validation, which is used to evaluate the performance of the proposed method. The second is the comparison with other works, such as Shao et al. [19] and Li et al. [6]. The third is the comparison of different receptive field sizes of CNN models. The last is the visual explanations of the most indicative regions of tooth-marked tongue using Grad-CAM. The experiments results are evaluated by the following five metrics: (1) Accuracy; (2) Precision; (3) Recall; (4) F1 Score; and (5) F2 Score. TP, FP, TN, and FN represent true positive, false positive, true negative, and false negative, respectively.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{6}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{7}$$

$$\text{F2 Score} = 5 \times \frac{\text{Precision} \times \text{Recall}}{4 \times \text{Precision} + \text{Recall}}. \tag{8}$$

### *4.1. Dataset*

As described in [14], the tongue image data should be collected in a uniform environment and contain as many high-quality images as possible. The dataset we used was provided by Shanghai Daosh Medical Technology Company, Ltd, Shanghai, China. It contained images taken at three different times for a total of 645 tongue images. These images were labeled by Chinese medicine experts. Among them, 346 nontooth-marked tongue images were marked as negative examples, and 299 tooth-marked tongue images were marked as positive examples [6].

### *4.2. Training*

We used the above dataset to train our CNN model, described in Section 3.2. Before inputting the images into the network, we adapted some images, preprocessing to separate the tongue body from the background, and downscaled the images to 256 × 256 and cropped them randomly to 224 × 224. Since each person's tongue color was slightly different, and the color of the tongue has little effect on the recognition of the tooth-marked tongue, we also augmented the training data with random horizontal flipping and brightness adjustments.

The network was trained end-to-end using Adam with standard parameters ($\beta_1$ = 0.9 and $\beta_2$ = 0.999) [29]. We trained the model using minibatches of size 16. We used an initial learning rate of 0.001, which was decayed by a factor of 0.8 following every 2000 epochs, and we stopped our training after 12,000 epochs, since the accuracy was basically stable beyond this point.

### *4.3. Test*

Five-fold cross-validation was used to test the proposed method. Since our dataset included a total of 645 tongue images obtained at three different times, the tongue images were randomly divided into five groups, each group containing 129 tongue images. Each time, four groups were used for training and the other one was used for testing. Table 1 shows the results of each cross-validation experiment. The proposed method was relatively stable and the average classification accuracy reached 78.6%.

**Table 1.** Five-fold cross-validation results.

| | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **F2 Score** |
|---|---|---|---|---|---|
| Fold_1 | 76.0% | 74.2% | 73.8% | 0.74 | 0.74 |
| Fold_2 | 77.5% | 74.4%. | 75.6% | 0.75 | 0.76 |
| Fold_3 | 79.8% | 79.0% | 79.0% | 0.79 | 0.79 |
| Fold_4 | 82.2% | 84.1%. | 80.0% | 0.82 | 0.81 |
| Fold_5 | 77.5% | 70.7%. | 82.1% | 0.76 | 0.80 |
| Average | 78.6% | 76.5% | 78.1% | 0.77 | 0.78 |

### *4.4. Comparison*

We conducted experiments on our dataset using another three methods, mentioned in Section 2.1, which were proposed by Wang [18], Shao [19], and Li [6]. The cross-validation settings used in these experiments were the same as in Section 4.3. The average accuracy and recall of these four methods are recorded in Figure 3.

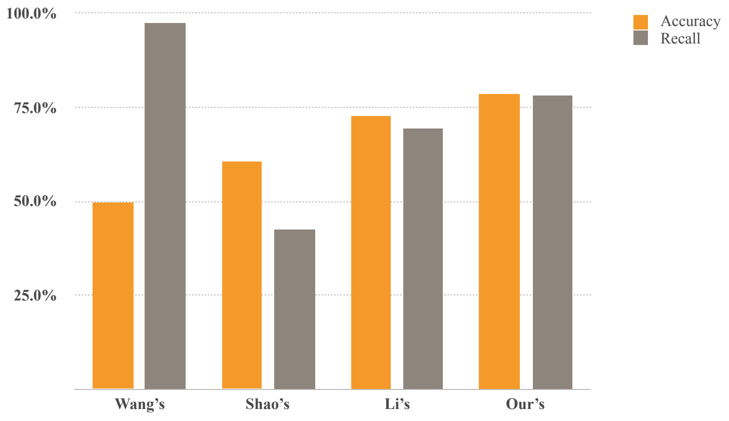


**Figure 3.** Comparison with other tooth-marked classification methods. Wang and Shao set thresholds based on concavity information, while Li's and our methods extracted features using CNN. Orange bars represent the accuracy and gray bars represent the recall of these four methods.

Most of traditional methods are designed based on the experience and ideas of the researchers. They are highly interpretable but not very accurate (or robust). As we can see from the results, though Wang's method had a high recall, it only used concavity information; which could easily misjudge the concave regions on the healthy tongue. Thus, the overall accuracy was low. Shao's method effectively improved the accuracy, but the recall is not guaranteed. These two methods both extract features manually and set thresholds that match the specific dataset. They don't have good generalization ability and fail to achieve a good balance between accuracy and recall. Li extracts tooth-marked tongue features using a VGG16 model, while we extract features using the model described in Section 3.2. For a more detailed comparison of these two methods, we provide the receiver operating characteristic

(ROC) curves in Figure 4. We also calculated the Area Under Curve (AUC) values of these two methods. Li's method had an AUC of 0.81 and our method had an AUC of 0.85. These two methods both had stable performance, while our model used a shallower network and achieved better results.

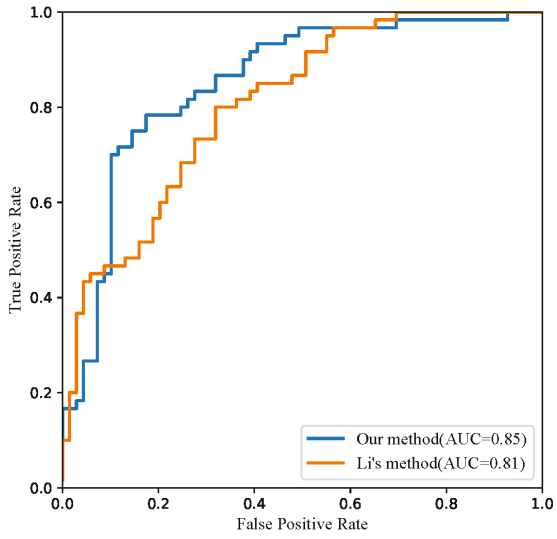


**Figure 4.** ROC curves for Li's method and our method. Li's method has an AUC of 0.81 and our method has an AUC of 0.85.

We provide two examples, showing the explanations from Li's method and ours, in Figure 5. The left column is the original tongue image and the tooth-marked regions are provided by TCM practitioners which are marked by blue boxes. The middle column is the visualizations provided by Li's method and the right column is the visualizations provided by our method. Although both of the methods classify the tongue image correctly, by generating Grad-CAM visualizations, we find that these two models are different in their attention. In the first row, the region on the left of the tongue has deceived Li's method, but in our method, the true region of the teeth mark is successfully locked down. In the second row, our method can correctly find the regions of the teeth mark and ignore other irrelevant regions. Li's method not only highlights the tooth-marked region but also highlights the non-toothed regions, such as the upper part of the tongue. Non-toothed regions are given a high weight, meaning that the model does not focus on the regions that have the greatest impact on the classification, which is why our method's classification accuracy is higher.

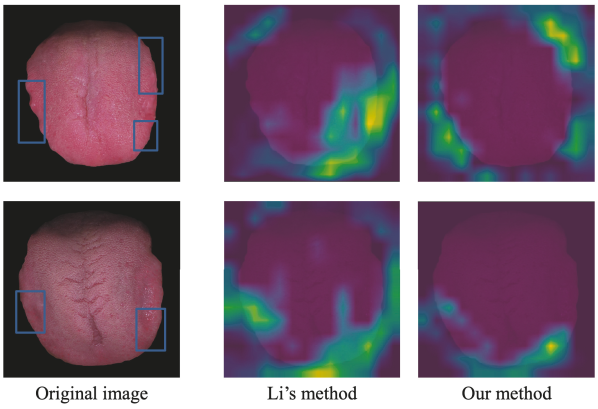


**Figure 5.** Grad-CAM explanations for Li's method and our method. We can see that, even though both methods made the right decision, these two models are different in their attention.

## *4.5. Effects of Parameters*

Several parameters are involved in our CNN model design. In this section, we examine how these parameters affect the network performance. Figure 6 shows how the performance varies with respect to the number of convolution (Conv) kernels. The network architecture starts from a 256-kernel in 2-Conv layer to 5-Conv layer and halving parameters. We find that, although many mainstream networks use 256 or more kernels every layer, our model tends to prefer 128 kernels. This may be because the background of the tongue image is single, the position of the tongue is clear, and too many parameters will lead to under-fitting of the model.

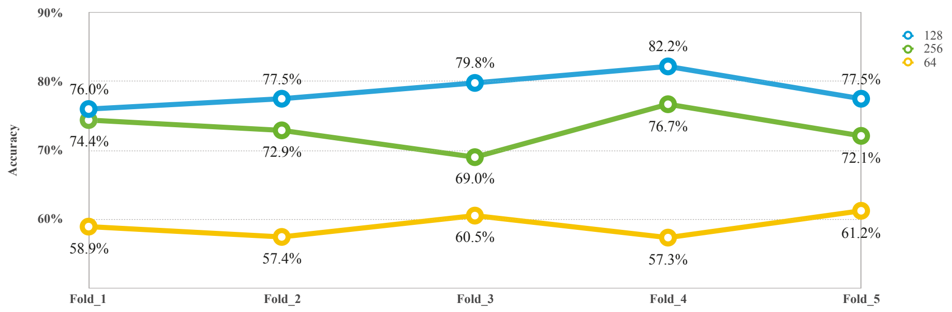


**Figure 6.** Accuracy with respect to varying the different number of convolutional kernels. The blue, green, and yellow lines represent 128, 256, and 64 kernels, respectively.

Through observation, we found that the size of the tooth mark region accounts for about 1/8 of the whole picture. Therefore, in theory, the neuron receptive field for detecting tooth marks doesn't need to be too large. For this reason, we explored the influence of different sizes of receptive fields on the classification results. The receptive field was defined as the region in the input space that a particular CNN's feature is looking at [30]. Our choices for kernel size include (3 × 3, 5 × 5, and 7 × 7), and for convolutional layers include (3, 4, 5, and 6). Based on [30], we compute the size of receptive fields in different network models. The experiment results are shown in Tables 2 and 3. We find that the model with a 3 × 3 kernel size is the most effective. As stated in [31], 3 × 3 kernel layers have more non-linear rectifications, which makes the decision function more discriminative. A 5-Conv layer whose receptive field is 94 performs better than others. In [9], it was found that the best-looking visualizations are often obtained after the deepest convolutional layer in the network, and localizations get progressively worse at shallower layers. This is because the later convolutional layers capture high-level semantic information and retain spatial information, while the shallower layers have smaller receptive fields and only concentrate on local features. However, in our research, we find that a 5-Conv layer network performs better than a 6-Conv layer network. So, the accuracy of image classification does not depend entirely on the number of network layers, but also on the specific classification tasks. It is necessary to understand the problem deeply and build a suitable network model to solve the specific problem.

**Table 2.** Comparison between different kernel sizes with the same convolutional layers, and the 3 × 3 kernel size is the most effective. (Conv Layer: 5-Conv layer).

| Kernel Size | Receptive Field | Accuracy | Precision | Recall | F1 Score | F2 Score |
|---|---|---|---|---|---|---|
| **3 × 3** | **94** | **78.6%** | **75.9%** | **78.1%** | **0.77** | **0.78** |
| 5 × 5 | 156 | 76.0% | 74.1% | 74.1% | 0.74 | 0.74 |
| 7 × 7 | 218 | 73.6% | 71.1% | 72.9% | 0.72 | 0.73 |

**Table 3.** Comparison between different convolutional layers with the same kernel size, and the 5-Conv layer is the most effective. (Kernel Size: 3 × 3).

| Conv Layer | Receptive Field | Accuracy | Precision | Recall | F1 Score | F2 Score |
|---|---|---|---|---|---|---|
| 3-Conv | 22 | 71.3% | 70.4%. | 62.1% | 0.66 | 0.64 |
| 4-Conv | 46 | 73.6% | 77.3%. | 56.1% | 0.65 | 0.59 |
| **5-Conv** | **94** | **78.6%** | **75.9%** | **78.1%** | **0.77** | **0.78** |
| 6-Conv | 190 | 72.1% | 72.5%. | 67.7% | 0.70 | 0.69 |

## *4.6. Model Interpretation*

To interpret the model predictions, we analyze how the localizations change qualitatively as we perform Grad-CAM with respect to different features maps in our model. As we can see from Figure 7, in the first few layers, CNN pays more attention to the edge and color information which are important for the next layers. Then, later convolutional layers start to detect the texture associated with the tooth marks. The Grad-CAM highlights the discriminative regions, which are usually some indentations along the lateral borders, and the color and brightness of them differ from the normal regions. It is interesting to see how our recognition method can serve as a tool to understand the network better by providing a localized high-resolution visualization of the tooth-marked regions. It shows precise localization to support the model's prediction. We think it helps to judge which model is more effective, as stated in Section 4.4, and also helps doctors to analyze the region of the tooth-marked tongue, rather than simply giving the classification results.

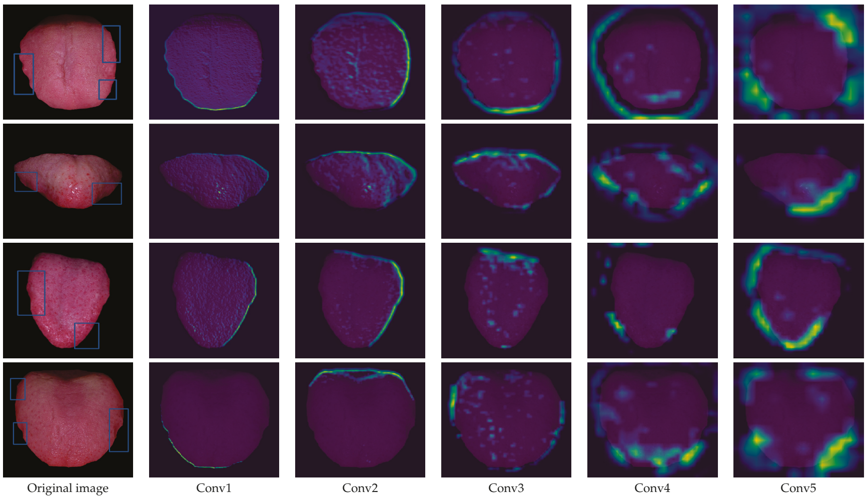


**Figure 7.** Grad-CAM localizations for the "tooth-marked tongue" category on different convolutional layer feature maps in our model. The first column is the original tongue image and teeth-marked regions are contained in blue boxes. The remaining columns of each row corresponds to Conv1–Conv5.

## 5. Conclusions

Automated recognition of tooth-marked tongue would not only have benefit in clinical settings, but it would also be invaluable in the delivery of health care to populations with inadequate access to diagnostic imaging specialists. In this paper, we have presented a tooth-marked tongue recognition method based on deep features and localized discriminative regions using Grad-CAM. The experiments showed that the proposed method has greatly improved accuracy, compared to previous methods. We mainly improve the problem in the following three aspects: (1) Analysis of the entire tongue picture—no need to cut small patches. (2) Analyzing the influence of different receptive field sizes on the classification results and finding the receptive field size suitable for the tooth-marked tongue problem. (3) Enhancing the interpretability of the CNN algorithm. Future work includes two aspects: (1) More new tongue samples will be acquired. Since we use a deep CNN as a feature extractor, the proposed model will benefit a lot from a larger dataset. (2) We will improve the architecture of the CNN model to further improve the accuracy and reduce the computation cost, and use patient history (or other relevant characteristics) in the future.

**Author Contributions:** Methodology, Experimental analysis and Paper Writing, Y.S.; Writing-review and Data analysis, S.D.; Data and Writing Correction, Y.Z. and J.L.; The work was done under the supervision and guidance of X.L.; All the authors revised the manuscript.

**Funding:** This research received no external funding.

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

## References

1. Shen, Z.Y. Basic theory of traditional Chinese medicine. *Chin. J. Integr. Tradit. West. Med.* **1997**, *17*, 643.
2. Pang, B.; Zhang, D.; Wang, K. Tongue image analysis for appendicitis diagnosis. *Inf. Sci.* **2005**, *175*, 160–176. [CrossRef]
3. McLean, N. Color atlas of oral diseases. *Br. J. Plast. Surg.* **2004**, *100*, 1299–1300. [CrossRef]
4. Li, W.; Luo, J.; Hu, S.; Xu, J.; Zhang, Z. Towards the Objectification of Tongue Diagnosis: the Degree of Tooth-marked. In Proceedings of the IEEE International Symposium on It in Medicine and Education, Xiamen, China, 12–14 December 2008; pp. 592–595.

5. Ren, Y.; Rong, L.; Ying, Z. Study on the correlation between dental scar tongue and constitution of traditional Chinese medicine in physical examination population. *World Sci. Technol.-Mod. Tradit. Chin. Med.* **2012**, *14*, 2283–2289.
6. Li, X.; Yin, Z.; Cui, Q.; Yi, X.; Yi, Z. Tooth-Marked Tongue Recognition Using Multiple Instance Learning and CNN Features. *IEEE Trans. Cybern.* **2019**, *49*, 380–387. [CrossRef] [PubMed]
7. Zhang, D.; Pang, B.; Li, N.; Wang, K.; Zhang, H. Computerized diagnosis from tongue appearance using quantitative feature classification. *Am. J. Chin. Med.* **2005**, *33*, 859–866. [CrossRef] [PubMed]
8. Wang, Y.G.; Yang, J.; Zhou, Y.; Wang, Y.Z. Region partition and feature matching based color recognition of tongue image. *Pattern Recognit. Lett.* **2007**, *28*, 11–19. [CrossRef]
9. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In Proceedings of the International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.
10. Chiu, C.C. A novel approach based on computerized image analysis for traditional Chinese medical diagnosis of the tongue. *Comput. Methods Progr. Biomed.* **2000**, *61*, 77–89. [CrossRef]
11. Pang, B.; Zhang, D.; Li, N.; Wang, K. Computerized tongue diagnosis based on Bayesian networks. *IEEE Trans. Biomed. Eng.* **2004**, *51*, 1803–1810. [CrossRef] [PubMed]
12. Zuo, W.; Wang, K.; Zhang, D.; Zhang, H. Combination of polar edge detection and active contour model for automated tongue segmentation. In Proceedings of the International Conference on Image and Graphics, Hong Kong, China, 18–20 December 2004; pp. 270–273.
13. Yu, S.; Yang, J.; Wang, Y.; Zhang, Y. Color Active Contour Models Based Tongue Segmentation in Traditional Chinese Medicine. In Proceedings of the International Conference on Bioinformatics and Biomedical Engineering, Wuhan, China, 6–8 July 2007; pp. 1065–1068.
14. Wang, X.; Zhang, B.; Yang, Z.; Wang, H.; Zhang, D. Statistical analysis of tongue images for feature extraction and diagnostics. *IEEE Trans. Image Process.* **2013**, *22*, 5336–5347. [CrossRef] [PubMed]
15. Huang, B.; Wu, J.; Zhang, D.; Li, N. Tongue shape classification by geometric features. *Inf. Sci.* **2010**, *180*, 312–324. [CrossRef]
16. Zhang, Y. Research on Analysis Method of Tongue and Teeth-Marked Tongue. Ph.D. Thesis, Beijing University of Chinese Medicine, Beijing, China, 2005.
17. Li, J.F.; Li, N.M.; Wang, K.Q.; Zhang, H.Z. Extracting feature of teeth-marked tongue image. In Proceedings of the Diagnosis Section of China Society of Integrated Traditional Chinese and Western Medicine, Fuzhou, China, 1 July 2009; pp. 100–105.
18. Wang, H.; Zhang, X.; Cai, Y. Research on Teeth Marks Recognition in Tongue Image. In Proceedings of the Academic Conference on National Diagnosis of Chinese Society of Integrated Traditional Chinese and Western Medicine, Shenzhen, China, 30 May–1 June 2014; pp. 80–84.
19. Shao, Q.; Li, X.; Fu, Z. Recognition of teeth-marked tongue based on gradient of concave region. In Proceedings of the International Conference on Audio, Language and Image Processing, Shanghai, China, 7–9 July 2015; pp. 968–972.
20. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
21. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
22. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 818–833.
23. Springenberg, J.T.; Dosovitskiy, A.; Brox, T.; Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv* **2014**, arXiv:1412.6806.
24. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning Deep Features for Discriminative Localization. In Proceedings of the Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27 June–30 June 2016; pp. 2921–2929.
25. Mahendran, A.; Vedaldi, A. Understanding deep image representations by inverting them. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 5188–5196.

26. Esteva, A.; Kuprel, B.; Novoa, R.A.; Ko, J.; Swetter, S.M.; Blau, H.M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **2017**, *542*, 115–118. [CrossRef] [PubMed]
27. Rajpurkar, P.; Irvin, J.; Zhu, K.; Yang, B.; Mehta, H.; Duan, T.; Ding, D.; Bagul, A.; Langlotz, C.P.; Shpanskaya, K.; et al. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv* **2017**, arXiv:1711.05225.
28. Mahendran, A.; Vedaldi, A. Visualizing Deep Convolutional Neural Networks Using Natural Pre-images. *Int. J. Comput. Vis.* **2016**, *120*, 233–255. [CrossRef]
29. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
30. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
31. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.

*future internet*

Article

# 3D-CNN-Based Fused Feature Maps with LSTM Applied to Action Recognition

**Sheeraz Arif *, Jing Wang *, Tehseen Ul Hassan and Zesong Fei**

Information and Communication Engineering, Beijing Institute of Technology, Beijing 100081, China; tehseen@bit.edu.cn (T.U.H.); feizesong@bit.edu.cn; (Z.F.)
* Correspondence: Sheeraz.arif@bit.edu.cn (S.A.); wangjing@bit.edu.cn (J.W.)

Received: 20 December 2018; Accepted: 6 February 2019; Published: 13 February 2019

**Abstract:** Human activity recognition is an active field of research in computer vision with numerous applications. Recently, deep convolutional networks and recurrent neural networks (RNN) have received increasing attention in multimedia studies, and have yielded state-of-the-art results. In this research work, we propose a new framework which intelligently combines 3D-CNN and LSTM networks. First, we integrate discriminative information from a video into a map called a 'motion map' by using a deep 3-dimensional convolutional network (C3D). A motion map and the next video frame can be integrated into a new motion map, and this technique can be trained by increasing the training video length iteratively; then, the final acquired network can be used for generating the motion map of the whole video. Next, a linear weighted fusion scheme is used to fuse the network feature maps into spatio-temporal features. Finally, we use a Long-Short-Term-Memory (LSTM) encoder-decoder for final predictions. This method is simple to implement and retains discriminative and dynamic information. The improved results on benchmark public datasets prove the effectiveness and practicability of the proposed method.

**Keywords:** action recognition; fused features; 3D convolution neural network; motion map; long short-term-memory

## 1. Introduction

Human activity recognition (HAR) is one of the enabling technologies behind human-computer interactions, video surveillance and video scene understanding [1]. To date, it imposes significant challenges such as the frequent presence of background clutter, view point changes, irregular motion, intra-class variations and camera motion. In addition, the huge information redundancy in video requires large amounts of memory, and also, the discovery of discriminative information from video frames is very complex and slow process.

The result of various research studies indicates that the success of action recognition problems depends on an appropriate feature extraction process. The appropriate feature extraction is very important in distinguishing samples and variations in the frames. Considerable progress has been made to address this problem by employing various specific solutions. Many local space-time visual representations have been proposed to overcome these issues in action recognition tasks. Laptev [2] detected sparse-time interest points and computed a histogram of the detected local points. Hessian [3], local trinary patterns (LTP) [4], Cuboids [5], and 3-D SIFT [6]) have also shown promising levels of HAR effectiveness, mainly thanks to their robustness against partial occlusions and noise. To facilitate a more effective usage of motion information, many trajectory-based feature extraction approaches have been proposed, such as KLT-tracker [7], SIFT matching [8], DTF [9], improved DTF [10]. However, there are number of weaknesses in these models, such as the presence of irrelevant and redundant trajectories, computational complexity and blending of unnecessary motion.

The ideal video representation method must be efficient to compute and simple to implement instead of using complicated and labor-intensive feature extraction and encoding methods. The extraction of spatiotemporal features from video frame sequences is widely used for recognizing human actions. Due to the advancement of digital camera technology, it has become possible to capture depth information, which can be embodied into a single motion map. Compared to dynamic and conventional images, motion maps can provide 3D information, and can be insensitive to changes in light conditions. Much research efforts has used depth imagery such as dynamic images [11] and depth maps [12] in the context of action recognition. These methods are able to process temporal information, but are insufficient to capture dense and discriminative information in terms of shape, appearance and motion.

Recently, deep convolutional neural networks (DCNNs) and Long Recurrent Convolutional networks (LRCNs) have shown great potential in many areas, and have yielded promising results for many computer vision tasks. These approaches have the ability to accurately identify the hidden pattern in visual data by back propagation, so features are auto-extracted without any artificial selection. It has been proven empirically that features learned from deep neural networks are much better than hand-crafted features.

In light of the above analysis, this research article examines the issue of human action recognition by using motion maps and intelligently incorporating a C3D network with a Long Recurrent Convolutional network (LRCN) network. We utilize a 3D convolutional neural network (C3D) [13] to acquire and integrate the temporal information. The C3D can model appearance and motion information simultaneously. Our model integrates a motion map of the previous frames with the next frame to generate a new motion map. We can get a motion map of the whole video after the repetitive integration of the next frame for various-length videos. We use a linear weighted fusion method to fuse feature maps to take advantage of spatiotemporal features. Finally, we use LSTM for feature encoding and action classification. The proposed method is simple to implement and acquires temporal information effectively, integrating it into a map without losing the discriminative information of videos. The proposed method shows significantly improved results over some baseline methods when applied to the various benchmark video datasets. It is worth highlighting the following contributions:

1. We propose an iterative training method for our neural network to generate a motion map from input video, which can integrate information into a motion map from each video frame.
2. We intelligently incorporate C3D and LSTM networks and capture long-range spatial and temporal dynamics. C3D features on video shots contain richer motion information; LSTM can explore the temporal relationship between video shots.
3. We introduce an effective fusion technique i.e., a linear weighted fusion method which can fuse correspondence between spatial and temporal features and boost recognition accuracy.
4. The effectiveness of our approach is evaluated on benchmark datasets, in which it obtained state-of-the-art recognition results.

The remainder of this article is organized as follows: Section 2 reviews related works. In Section 3, we present our proposed approach in detail. We demonstrate the experimental evaluation in Section 4. Finally, a conclusion is presented in Section 5.

## 2. Related Work

Over the last decade, researchers have presented many hand-crafted and deep-net-based methods for action recognition. Earlier works were based on hand-crafted features for non-realistic actions videos. Since the proposed method is based on deep neural network (DNN), in this section, we will only review related works based on DNN.

In recent years, different variants of deep learning models have been proposed for human activity recognition in videos, and have achieved great performance for computer vision tasks. Ji et al. [14]

applied 3D convolutional kernels on video frames in a time axis to capture both spatial and temporal information. Karpathy et al. [15] directly applied CNNs to multiple frames in each sequence and obtained the temporal relations by pooling, using single, late, early and slow fusion; however, the results of this scheme were just marginally better than those of a single frame baseline. Simonyan and Zisserman [16] used a two-stream CNN framework to incorporate both feature types, with one stream taking RGB image frames as the input and the other taking pre-computed stacked optical flows. Since optical flow contains only short-term motion information, adding it does not enable CNNs to learn long-term motion transitions. The additional stream significantly improved action recognition accuracy, indicating the importance of motion features. Tran et al. [13] avoided the need for pre-computing optical flow features through their 3D convolution (C3D) framework, which allows deep networks to learn temporal features in an end-to-end manner. However, C3D only covers a short range of the sequence. Wang et al. [17] introduced a temporal segment network (TSN) architecture, where a sparse temporal sampling strategy is adopted to model long-term temporal structures. In [18], Feichtenhofer et al. study a number of ways of fusing CNN towers in order to take advantage of this spatial-temporal information from the appearance and optical flow networks. However, the CNN-based method only extracts visual appearance features, and lacks the long-range temporal modeling capabilities. Moreover, the CNN-based method ignores the intrinsic difference between spatial and temporal domains.

Some researchers have also presented methods by uniting the benefits of both hand-crafted and deep learned features, such as [19,20], and obtained good results. They integrate the key factors from two successful video representations, namely improved trajectories [10] and two-stream ConvNets [18]. How to combine the benefits of these two kinds of features to design good descriptors has been an active research area. Some research efforts have been carried out using depth imagery such as dynamic images and depth maps. Bilen et al. [11] introduced the dynamic image network to generate dynamic images for action videos. The order of video frames is used as the supervisory information; however, this method loses some discrimination information. Chen et al. [12] represented a model in the form of depth maps in the context of action recognition. These contributions showed good action recognition results but were insufficient to capture dense and discriminative information in terms of shape, appearance and motion. Taylor et al. [21] used a convolutional gated restricted Boltzmann machine to generate a flow field of the adjacent two frames in the video for action recognition, but this model could not generate a single map to represent a video. Rank pooling [22] and Fisher Vector [23] made an attempt to generate the desire length motion map. However, these methods are unable to model temporal dynamics among video frames.

In order to model the temporal dynamics among video frames, RNNs have been considered for video-based HAR. RNN networks provide strength to find and process hidden patterns in time-space data. In these kind of systems, data is processed in a sequential way, such that at each time t, it gets input from the previous hidden state $s_{t-1}$ and obtains new data $x_t$. Most of the state-of-the-art methods [24–29] have proposed their own recurrent networks by leveraging CNNs and RNNs for action recognition, and have achieved impressive performance. However, due to the large number of calculations of parameters, and negligence of effect of initial input after few layers, vanishing gradient problems occurred. The solution to this problem is LSTM [25,27,30], which has the ability to capture long-term dependencies and preserve sequence information over time by integrating memory units. LSTM was first introduced by [31]; it has been successfully adapted to many sequential modelling tasks such as speech recognition, visual description and machine translation, and has achieved encouraging performance. In most of these networks, the inputs to the LSTM are the high level features captured from a fully-connected layer of CNN. LSTM units use multiplicative gates to control access to the error signal propagating through the networks.

In this paper, we propose a 3Dconv-based iterative training method to generate the motion map, enabling the use of existing CNN models directly on video data with fine-tuning. Our model efficiently integrates the temporal information of the motion map and video frames and generates the arbitrary length of the motion map. The Combination of CNN-RNN provides effective representation for long-term motion and modeling of the sequential data, each of which has a time relationship with adjacent points. (RNN uses the extracted C3D features as inputs and models more robust, longer-range features.) The C3D network is able to encode local temporal features within each video unit; it cannot model across the multiple units of a video sequence. We thus introduce LSTM to capture global sequence dependencies of the input video and cues on motion information. The fused spatio-temporal features are processed by LSTM, which helps recognizing complex frame-to-frame hidden sequential patterns. After conducting extensive experiments, we observed that our method is very effective for videos of various lengths, and shows significant improvement in action recognition.

## 3. The Proposed Approach

In this section, the proposed approach and its related components are discussed. The process of action recognition is divided into two parts: The first part is related to the extraction of spatiotemporal fused features, so we discuss this within the relevant subsequent sections, e.g., the generation of motion maps and the training of motion map networks. Finally, we explain the encoding of the extracted features and the action classification part in the main subsequent section.

### *3.1. Extraction of Spatio-Temporal Fused Features*

#### 3.1.1. Generation of Motion Map

A motion map is a powerful and compact representation of a video which can be useful in computer vision tasks. The motion map can visualize motion information in good manner, and can remove a large amount of information redundancy of the video, thereby revealing discriminative information. The calculation of the motion map is fast, and takes up fewer memory resources. Hence, using a map to represent the video has realistic requirements. Our propose model is very simple to implement and can be trained by increasing the training video length iteratively. Mainly, it is very helpful to solve the problem of videos of various lengths to get the same effect of the map representation, and also to integrate the temporal information into a map without losing the discriminative information of the video. Another advantage of this method is that we can extract a constant number of video frames per second, which improves the generalization performance of the network. We can utilize a 3D-convolutional neural network for the extraction of the motion map. 3D convolution and 3D pooling operations are adopted in 3D ConvNets. Three-dimensional convolution is the extension of 2D convolution. The output of the 2D convolution are two-dimensional feature maps, while the output volume of 3D convolution can have multiple-dimensions. Each feature map of the convolutional layer is connected with some successive adjacent frames in upper layer. As a result, the temporal information is not lost and the motion of the human body can be efficiently captured. Hence, multiple 3D convolutional layers can be used to handle the spatial and temporal information of the inputs in a hierarchal way.

For a video $V$ with $N$ frames, we define the video frames as $fi$, $i$ $\{1 \ldots \ldots N\}$. $F_i$ denotes the motion map from $f_1$ to $f_i$. In order to retain appearance and action information, we introduce an iterative method to generate a new motion map $F_{i+1}$ using Equation (1) by combining the current motion map $F_i$ with the future video frame $f_{i+1}$ by using MMN. Symbol $\oplus$ is the pixel-wise addition between motion map and video frame. The process of generating our first and final motion map is shown in Figure 1a,b respective

$$F_{i+1} = F_i \oplus f_{i+1} \tag{1}$$

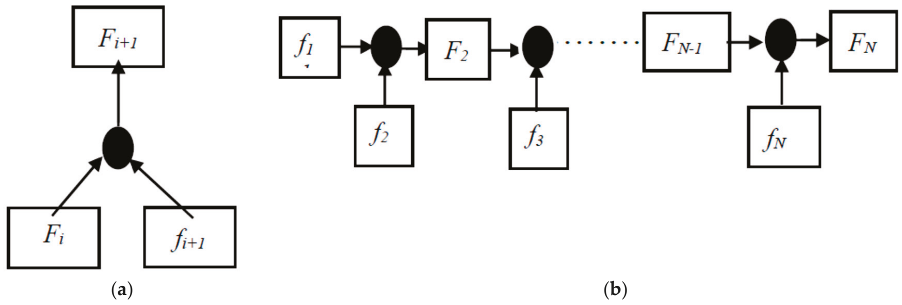


**Figure 1.** (**a**,**b**) Generation of our first and final Motion Map.

In the last iteration of our MMN network, we obtain the final motion map $F_N$ of video $V$, in which discriminative information is embodied and can be applied for action recognition tasks. Some motion maps generated by our C3D network are listed in Figure 2. Each map highlights the static object with its main features, and the superposed silhouette incarnates the different locations and postures of the actor and objects. For example, the action category playing violin shows that the actor, arms and violin are the main features, while the rest of the image is diluted. This shows the relationship between the arm movements and playing the violin. It reflects the motion relationship between the actor and the object, and proves that the dynamic information which is available in different sequences of the video can be retained and embodied in the motion map.

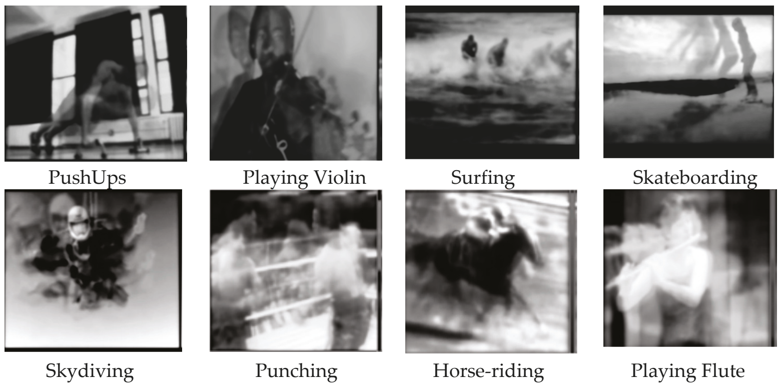


**Figure 2.** Output Motion map generated by our network, illustrating the discriminative information integrated into a single motion map to classify the video category.

### 3.1.2. C3D Network Architecture

The C3D network has the ability to learn visual patterns directly from pixels without any pre-processing step. The architecture of C3D comprises trainable filters and local pool operations, which is very useful to find hidden patterns in a video frame, and captures all changes in terms of spatial and temporal information.

The architecture of the C3D network is given in Figure 3. Table 1 illustrates the different parameter settings of each convolutional and pooling layer. We set the 3D Convolution and pooling kernel size as $d \times k \times k$, where $d$ is kernel temporal depth and $k$ is kernel spatial size. The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. By this construction, the feature maps in the convolution layer are connected to multiple contiguous frames in the previous layer, thereby capturing motion information. Intuitively, these different layers describe the visual content at different level, each of which is complementary to each other for the task of recognition. The C3D network has 5 convolution layers and 5 pooling layers (each convolution layer is immediately followed by a pooling layer), 2 fully connected layers and softmax loss layer. The number of channels (filters) for 5 convolution layers from 1 to 5 is 64, 128, 256, 512, and 512 respectively. The ratio represents the spatial map size ratio. In both spatial and temporal dimensions, all convolutional layers have $3 \times 3 \times 3$ convolution filters with stride $1 \times 1 \times 1$. All pooling layers from pool2 to pool5 (except for the first layer) have $2 \times 2 \times 2$ pooling kernels with stride $2 \times 2 \times 2$, which means the size of the output signal is reduced by a factor of 8 compared with input signal. The first pooling layer, i.e., the pool1 layer, has a kernel size of $1 \times 2 \times 2$, with the goal of not merging the temporal signal and preserving the temporal information in the early phases. The output of each convolution-al layer is a kind of volume in the form of feature maps. All pooling layers lead to the same number of feature maps as convolution layers but with reduced spatial resolution; also, these pooling layers introduce scale-invariant features. The two fully connected layers have 2048 outputs, and finally, a softmax layer is used to predict action labels.

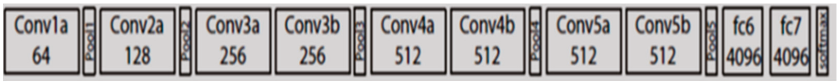


**Figure 3.** Complete Network architecture of C3D.

**Table 1.** The convolutional and pooling layers of the C3D architecture.

| **Layers** | **Conv1a** | **Conv2a** | **Conv3a** | **Conv3b** | **Conv4a** | **Conv4b** | **Conv5a** | **Conv5b** |
|---|---|---|---|---|---|---|---|---|
| Size | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ | $3 \times 3 \times 3$ |
| Stride | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ | $1 \times 1 \times 1$ |
| Channel | 64 | 128 | 256 | 256 | 512 | 512 | 512 | 512 |
| Ratio | 1 | 1/2 | 1/4 | 1/4 | 1/8 | 1/8 | 1/16 | 1/1 |
| **Layers** | **Pool1** | **Pool2** | **Pool3** | **Pool4** | **Pool5** | **Fc6** | **Fc7** | |
| Size | $1 \times 2 \times 2$ | $2 \times 2 \times 2$ | $2 \times 2 \times 2$ | $2 \times 2 \times 2$ | $2 \times 2 \times 2$ | - | - | |
| Stride | $1 \times 2 \times 2$ | $2 \times 2 \times 2$ | $2 \times 2 \times 2$ | $2 \times 2 \times 2$ | $2 \times 2 \times 2$ | - | - | Softmax |
| Channel | 64 | 128 | 256 | 512 | 512 | 4096 | 4096 | Layer |
| Ratio | 1/2 | 1/4 | 1/8 | 1/16 | 1/32 | - | - | |

Figure 4 illustrates the single iteration process of our Motion Map Network (MMN). The input to our network is frame-by-frame RGB clip. A motion map and the next video frame are combined into a video frame sequence as input, and a single 2D-feature map is extracted as output. At this stage, it is very important to mention that the feature maps are extracted in a frame-by-frame manner. We compute feature maps of layer conv5b from the input videos, and the rest of the pool5 and full-connected layers are abandoned in our scheme. C3D conv5b feature maps have the highest activation projected back to the image space. In each iteration, the output of the conv5b layer generates two feature maps, each with a size of $7 \times 7 \times 512$, where $7 \times 7$ is the spatial size of the feature maps with 512 channels. So, we build only one feature map of $7 \times 7 \times 512$ by taking the maximum value for each position of the both feature maps from conv5b. This process is applied for all iterations for our pipeline, except the last iteration, because the output of the last iteration is again with two feature maps. We will apply linear weighted fusion to the last two feature maps by taking advantage of spatial-temporal features to obtain our final feature map. The discriminative information embodied in the final motion map can be applied to human action recognition tasks.

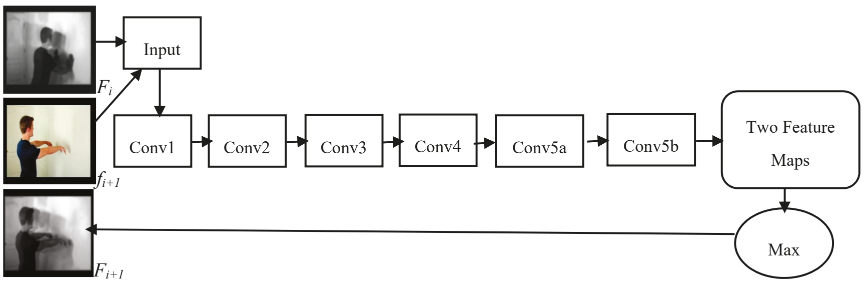


**Figure 4.** The structure of our Motion Map Network, which illustrates the single iteration to generate Motion Map.

### 3.1.3. Training of Motion Map Network (MMN)

Since the network can only handle video frames, the videos need to be processed as video frames. Some methods directly split the videos, ignoring different frame rates, so dynamic information may be inconsistent in time. Therefore, we extract a constant number video frames per second, which improves the generalization performance of the network. For the network to better capture the changes in the action, we extract two frames per second. As for some short videos, we loop the extracted frames, and fill up 16 frames per video.

We introduce an iterative method to train the Motion Map Network (MMN). We use video $V$ with $N$ frames and video labels $L$ to train our MMN. $S$ is defined as maximum training iteration length. We train MMN using training length $s$ from 2 to $S$. The training length $s$ is the round of iteration. We cut the training video $V_i$ into $s$-length clips $C_j^i$ $(j \in 1 \ldots .N_{\mathrm{i}}/s)$ with overlap 0.7 and assign the labels $L_{\mathrm{i}}$ to clip $C_j^i$. We define the MMN as a function $Z_{\theta_s}(I_a, I_b)$, where $\theta_s$ denotes the parameters of MMN after the iteration of training length $s$. The initial parameters of the network are defined as $\theta_1$. For each $s$, we generate the motion map $F_{1\sim s-1}^{c_j^i}$ using $Z_{\theta_{s-1}}$, and train the MMN using the motion map $F_{s-1}^{c_j^i}$, video frame $f_s^{c_j^i}$ and video clip label $L_i$. Finally, we can get $\theta_s$ which is the parameter of our trained motion map network. The detail of the training steps is summarized in Algorithm 1.

**Algorithm 1.** Training of Our Motion Map Network

```
**Input:** $V$ is Video dataset; Frame number of video dataset, $N$;
Video labels, $L$; Maximum training iteration length, $S$;
Parameters of our model, $\theta_1$;
**Output:** Final parameters of Network, $\theta_s$;
1: Initialize the parameter $\theta_1$ for our model;
2: *for* each $s \in 2,3,\ldots,S$ *do*
3: cut $V_i$ into $s$-length clips $C_i^j (j \in 1 \ldots N_{\mathrm{i}}/s)$ with overlap 0.7;
4: Extract the video frames from $C_i^j$ as $f^{C_i^j}$;
5: *for* each $j \in 1, 2 \ldots N/s$ *do*
6: *for* $k \in 1, 2 \ldots s-1$ *do*
7: Generate the motion map $F_k^{C_i^j}$ using $Z_{\theta_{s-1}}(F_k^{C_i^j} - 1, f_k^{C_i^j})$ *end for*
8: Train the MMN using $F_{s-1}^{C_i^j}, F_s^{C_i^j}$ and $L_{\mathrm{i}}$ *end for*
9: Get the MMN parameters $\theta_s$; *end for*
```

### 3.1.4. Fusion Method

The motion of the object can be observed via changes in both appearance and semantics. Based on this, we follow a feature fusion strategy to combine spatial and temporal information. Given, $X_s \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{T}}$, $X_t \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{T}}$ are the extracted frame level spatial and temporal features, where $H$ and $W$

are the height and width of the feature maps, $T$ is the number of frames. Before the fusion operation, we have to reshape both features maps (spatial and temporal) into vectors, which can be given as:

$$X = [X_s, X_t] \tag{2}$$

Now, we perform a pixel-wise addition which is known as linear weighted fusion between $X_s$ and $X_t$ to compute a single feature map $F$.

$$F = w_s F_s \oplus w_t F_t \tag{3}$$

where, $X \in \mathbb{R}^{\mathrm{H \times W \times T}}$, $\oplus$ is a matrix addition, $w_s$ and $w_t$ are weights of appearance and motion for spatial and temporal features maps, respectively. The weights are used to measure the significance of spatial and temporal features. After performing the fusion operation, we can define the new representative features as $x_{f,t}$ for the video clip. So, for the input video, a set of fused features ($x_{f,1}, \ldots \ldots, x_{f,t}, \ldots \ldots, x_{f,N}$) can be generated. Finally, we apply LSTM on these generated features to perform temporal encoding for human activity prediction.

## 3.2. *Encoding and Activity Classification*

This is the second and final part of our approach, which starts from detailed discussion on LSTM features and its architecture; then, we present the encoding and activity classification method.

### 3.2.1. Long Short-Term Memory (LSTM)

To analyze the hidden sequential patterns, it is natural choice to use RNN to encode the temporal structure of extracted sequential features. In video, visual information is represented in many frames which help in understanding the context of an action. RNN can interpret such sequences, but in cases of long term sequences, it usually forgets the earlier input sequence. LSTM has been designed to mitigate the vanishing problem and to learn long-term contextual information from temporal sequences. LSTM is a kind of recurrent network which can capture long-term dynamics, and which preserves sequence information over time. In addition, the LSTM gradient does not tend to vanish when trained with back propagation through time. Its special structure with input, output and, control gates control long-term sequence pattern identification. The gates are adjusted by a sigmoid unit that learns during training when to open and close. We adopt LSTM for encoding and decoding to recognize human actions.

The architecture of a LSTM unit is depicted in Figure 5. $x_t$, $c_t$, $h_t$ and $y_t$ stand for input vector, cell state, hidden state and output at the $t$-th state, respectively. The output $y_t$ depends on hidden state $h_t$, while $h_t$ depends on not only the cell state $c_t$, but also on its previous state. Intuitively, the LSTM has the capacity to read and write to its internal memory, and hence, to maintain and process information over time. The LSTM neuron contains an input gate $i_t$, a memory cell $c_t$, a forget gate $f_t$, and an output gate $o_t$. At each time step $t$, it can choose to write, read or reset the memory cell through these three gates. This strategy helps LSTM to access and memorize information in many steps. Equations (4)–(9) demonstrate the operation of temporal modelling performed in LSTM unit.

W and b are the parameters of the LSTM known as weights of the input vector and bias term. $S$ means a sigmoid function, *tanh* is the activation function and $\otimes$ is the element-wise multiplication. The cell state and output are computed step by step to extract long-term dependencies. The input to LSTM is $x_t$, which is the feature vector. A forget gate is used to clear the information from the memory unit, and an output gate keeps the information about the upcoming step. We also have $g_t$, which is computed from the input of the current frame and state of the previous state $h_{t\text{-}1}$. The hidden state of LSTM step is computed by using a *tanh* activation function and memory cell $c_t$.

$$i_t = S(w_{xi}\, x_t + W_{hi} h_{t-1} + b_i) \tag{4}$$

$$f_t = S\left(w_{xf}\, x_t + W_{hf} h_{t-1} + b_f\right) \tag{5}$$

$$o_t = S(w_{xo}\, x_t + W_{ho}h_{t-1} + b_o) \tag{6}$$

$$g_t = tanh(w_{xc}\, x_t + W_{hc}h_{t-1} + b_c) \tag{7}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \tag{8}$$

$$z_t = \; h_t = o_t \otimes tanh\;(c_t) \tag{9}$$

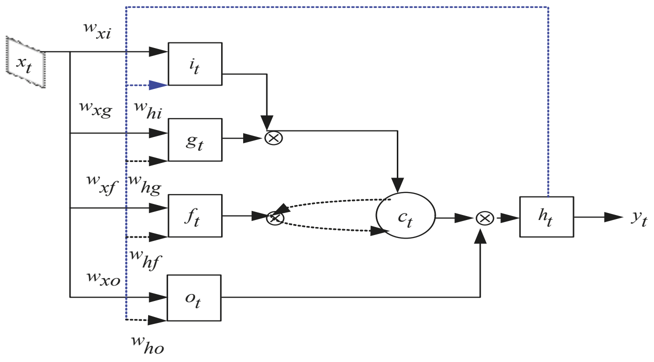


**Figure 5.** The architecture of LSTM Unit.

### 3.2.2. Encoding and Classification Process by LSTM

The generated fused features $x_{f,t}$ are fed into LSTM as inputs to conduct encoding and decoding for activity prediction. LSTM can be jointly trained, and our proposed model provides a trainable platform which is ideal for large-scale cognitive intelligence. The unique feature of LSTM is that it processes variable length inputs and produces high-level variable length predictions (Output). As shown in Figure 6, LSTM consists of an encoder and decoder; the encoder transforms input data $x_t$ to a corresponding activation $h$. The decoder in the output layer is trained to reconstruct an approximation $y$ of the input from activation $h$.

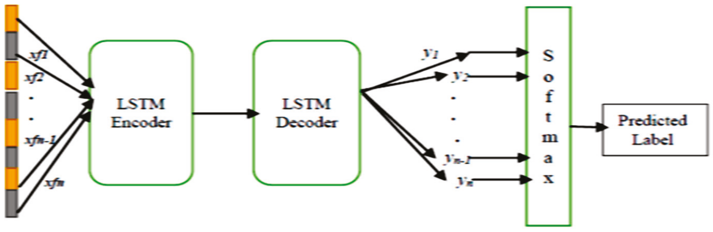


**Figure 6.** The framework of LSTM (Encoder-Decoder).

In general, the LSTM model has parameters i.e., $W$ and $b$, which denotes the weights and the biases of input layer and the hidden layer respectively, generates an output $z_t$ of given input $x_t$ and a previous hidden state at time step $t-1$ i.e., $h_{t-1}$, and also updates the current hidden state $h_t$.

$$z_t = S(W_1 x_t + b_1) \tag{10}$$

The next step is decoding, which is similar as the encoding step given in Equation (10), where $W_2$ and $b_2$ denotes the weights and biases of the hidden layer and the output layer.

$$y_t = (W_2 z_t + b_2) \tag{11}$$

The final single label prediction for a video can be produced by using softmax classifier. A Softmax layer can be utilized to achieve the M-way class scores for a given video sequence. This single prediction can be achieved by averaging the label probabilities, which is the output of our decoder, and can be represented by the Equation (11).

$$P(y^{q}_{(t)} = 1) = softmax(y_t) = softmax(W_{zt} + b_t) \tag{12}$$

where $W$ and $b_t$ are the trained parameters of the LSTM model, $q \in Q$ is the prediction and $t$ is the current time step.

## 4. Experiments

We conduct several experiments to validate the effectiveness of our system. Three well-known benchmark human action datasets, UCF101 [32], HDMB51 [33], and UCF Sports [34], have been used. The description of datasets with their validation schemes, experimental setup, results and comparative analysis are presented in subsequent sections.

### *4.1. Datasets*

The UCF101 dataset is the extension of UCF50; it contains 101 different action categories. Each action category consists of at least 100 video. There are 13,320 video clips in total. Most of the video clips are realistic, clean and user-uploaded videos with cluttered background, illumination and camera motion. The dataset is divided into a training set containing 9.5 K videos and testing set containing 3.8 K videos. We adopt the evaluation scheme of the THUMOS13 challenge [35] and follow the three testing/training split for performance evaluation by reporting average recognition accuracy over these three splits.

The HDMB51 dataset comprises of variety of realistic videos collected from YouTube and Google video. There are 6766 manually annotated video clips of 51 different action classes and each action class containing about 100 video clips. For experimental setting, we follow the original evaluation guidelines using three-test splits, and each split with an action class has 30 sequences for testing and 70 sequences for training. The average accuracy over these three splits is used to measure the final performance.

The UCF sports dataset encompasses 150 videos from 10 action classes. These videos were recorded in real sports environments, taken from different television channels. This dataset exhibiting the occlusion, illumination conditions and variations in background make it a complex and challenging dataset. The average accuracy is used to measure the final performance.

Some sample frames from three datasets are given in Figure 7.

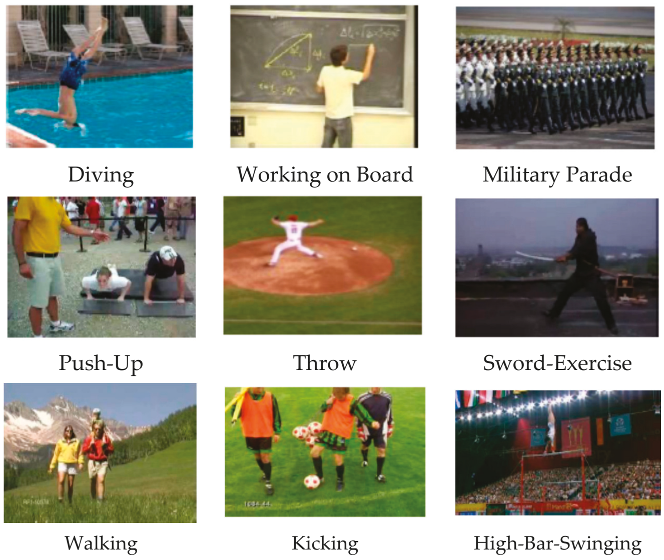


**Figure 7.** Sample frames from UCF101 (**first row**), HDMB51 (**second row**) and UCF Sports (**third row**).

### *4.2. Experimental Setup and Implementation Details*

As UCF101 is the largest dataset among the three datasets, we use it to train the C3D model initially, and then transfer it to the learnt model to HMDB51 and UCF sports for feature extraction. RGB clips are resized to have a frame size of 128 × 171. On training, we randomly crop input clips into 16 × 112 × 112 crops. We also horizontally flip them with 55% probability. We fine-tune the model parameters on the UCF101 dataset, where the initial learning rate is set as 0.003, which is divided by 2 every 150 K iterations. The optimization is stopped at 1.9 M iterations.

Since the network can only handle video frames, the videos need to be processed as video frames. Therefore, we extract a constant number video frames per second, which improves the generalization performance of the network. For the network to better capture the changes in the action, we extract two frames per second (fps) and loop the extracted frames, and fill up to 16 frames per video.

### *4.3. Results and Comparison Analysis*

We conduct extensive experiments to evaluate the performance of our proposed method. In this section, we presented relevant experimental results and performance analysis.

#### 4.3.1. Effect of Different Feature Fusion Techniques

In this section, we analyze the effect of different early fusion methods such as element-wise sum, concatenation, element-wise max and linear weighted fusion on our proposed framework. We show the recognition accuracy for UCF Sports dataset and also each split of UCF 101 dataset, and the average recognition accuracy over the three splits. The results are reported in Table 2. We observe that linear weighted fusion enhances the recognition accuracy of our approach by a fair margin, compared to other fusion methods. This enhancement may be due to the fact that the linear weighted fusion method efficiently fuses spatial and motion features. Therefore, we choose the linear weighted fusion method as our fusion scheme to fuse spatio-temporal features.

**Table 2.** Effect of different earlier fusion methods on our model. The accuracy (%) is computed on a UCF Sports dataset, and all three splits and their average on UCF101.

| Fusion Method | UCF Sports | Split 1 (UCF 101) | Split 2 (UCF 101) | Split 3 (UCF 101) | Average (UCF 101) |
|---|---|---|---|---|---|
| Element-wise max | 91.8 | 90.5 | 89.9 | 89.6 | 90.0 |
| Element-wise sum | 92.1 | 90.8 | 90.1 | 89.9 | 90.2 |
| Concatenation | 92.8 | 91.2 | 90.5 | 91.0 | 90.9 |
| Linear weighted | 93.9 | 91.6 | 90.9 | 91.7 | 91.4 |

### 4.3.2. Class-Wise Accuracy for Activity Recognition

This section computes the class-wise accuracy for action recognition. We investigate the recognition accuracy of our method by making a confusion matrix and considering 10 action classes of the UCF Sports dataset. Table 3 demonstrates the accuracy of each action category, the x-axis denotes the predicted labels and the y-axis represents the ground truth labels. The intensity of the true score is high (diagonal) for each category, and our method achieves 94% for all 10 classes. It is interesting to note that some of categories with similar actions are more easily confused with each other, such as golf swing, kicking, running, swing bench and walking; these categories interfere with each other and yield low scores. A possible reason for this is the similarity of the features and representations among actions. In addition, the number of training samples is too small, so the result is confusing and misclassification occurs. The confusion matrix of HMDB51 dataset is shown in Figure 8, which is well diagonalized. However, some categories are easily misclassified; nonetheless, our proposed approach still performs well with most action categories.

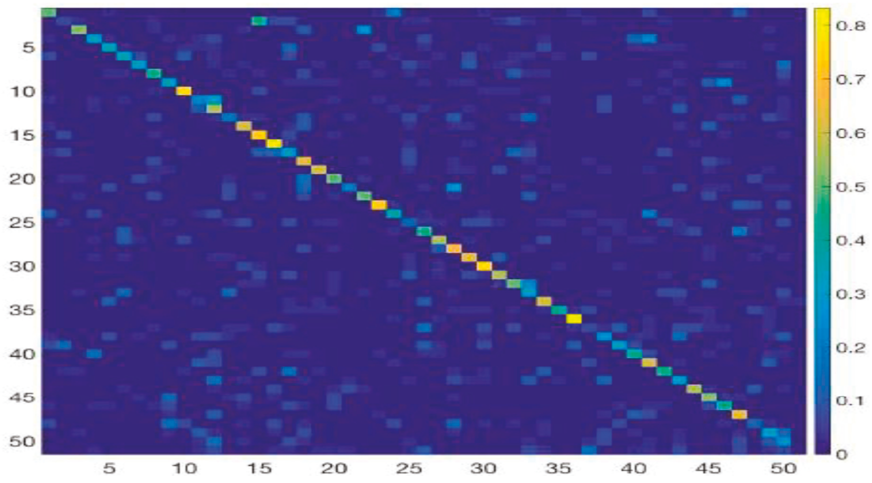


**Figure 8.** Confusion matrix on the HMDB51 dataset using our model.

**Table 3.** Confusion matric of UCF sports dataset.

| Categories | Diving | Golf-Swing | Kicking | Lifting | Horse Riding | Running | Skate Boarding | Swing Bench | Swing-Side | Walking |
|---|---|---|---|---|---|---|---|---|---|---|
| Diving | 1.00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Golf-Swing | 0 | 0.91 | 0.07 | 0 | 0 | 0 | 0 | 0.02 | 0 | 0 |
| Kicking | 0 | 0.06 | 0.94 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Lifting | 0 | 0 | 0 | 0.95 | 0 | 0 | 0 | 0 | 0 | 0 |
| Riding Horse | 0 | 0 | 0 | 0 | 0.90 | 0 | 0 | 0 | 0 | 0 |
| Running | 0 | 0.06 | 0.01 | 0 | 0.01 | 0.91 | 0 | 0 | 0 | 0.01 |
| Skateboarding | 0 | 0 | 0 | 0 | 0 | 0 | 0.93 | 0 | 0 | 0 |
| Swing Bench | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.00 | 0 | 0 |
| Swing Side | 0 | 0.01 | 0 | 0 | 0 | 0 | 0 | 0 | 0.99 | 0 |
| Walking | 0 | 0.07 | 0 | 0 | 0 | 0 | 0 | 0.04 | 0 | 0.89 |
| **Average Accuracy** | | | | | | | | | | 0.94 |

#### 4.3.3. Comparison to the State-Of-The-Art Methods

In this section, we further verify the effectiveness of our model, and compare our proposed approach to different existing state-of-the-art Human Action Recognition approaches on UCF101 and HDMB51 benchmark datasets. The comparison of results is reported in Table 4. We organize these baseline methods into different categories with respect to the type of features and network being used, including traditional, deep-learned features, very deep-learned features and hybrid features.

**Table 4.** Comparison to the state-of-the-art methods.

| Modality | Method | Year | UCF101 | HDMB51 |
|---|---|---|---|---|
| **Traditional** | iDT+fisher vector [10] | 2013 | 84.7 | 57.2 |
| | Ordered Trajectory [36] | 2015 | 72.8 | 47.3 |
| | MPR [37] | 2015 | - | 65.5 |
| | MoFAP [38] | 2016 | 88.3 | 61.7 |
| | Trajectory Rejection [39] | 2016 | 85.7 | 58.9 |
| **Deep** | Two-Stream [16] | 2013 | 88.9 | 59.4 |
| | FSTCN [40] | 2015 | 88.1 | 59.1 |
| | EMV-CNN [41] | 2016 | 86.4 | - |
| | DANN [42] | 2016 | 89.2 | 63.3 |
| | Dynamic Images [11] | 2016 | 89.1 | 65.2 |
| | LTC-CNN [43] | 2018 | 92.7 | 67.2 |
| **Very deep** | C3D [13] | 2015 | 85.2 | - |
| | LSTM [27] | 2015 | 88.6 | - |
| | LRCN [25] | 2015 | 82.9 | - |
| | VideoLSTM [44] | 2016 | 89.2 | 56.4 |
| | 3D Convolution [14] | 2016 | 91.8 | 64.6 |
| | STPP-LSTM [45] | 2017 | 91.6 | 69.0 |
| | FCNs-16 [46] | 2017 | 90.5 | 63.4 |
| | Hidden-Two-stream [47] | 2017 | 90.3 | 58.9 |
| | Multi-LSTM [48] | 2018 | 90.8 | - |
| **Hybrid-Model** | TDD-iDT [19] | 2015 | 91.5 | 65.9 |
| | C3D-iDT [13] | 2015 | 90.4 | - |
| | TSN [17] | 2016 | 94.2 | 69.4 |
| | 3D conv + iDT [14] | 2016 | 93.5 | 69.2 |
| | SCLSTM [49] | 2017 | 84.0 | 55.1 |
| | LTC-iDT [43] | 2018 | 92.7 | 67.2 |
| **Ours** | LSTM–3D ConvNet | - | 92.9 | 70.1 |

Compared to traditional methods, our model performs the best by 4.5% on both datasets, Compared with RNN-based methods such as (LRCN) [25] and (LSTM) [27], our model outperforms these two methods by 4.3% and 10% on UCF101 datasets respectively. Different experiments indicated that our approach possesses higher discriminative power, even using fewer parameters. It can be also seen that some methods with both features such as TSN [17] and 3D conv—iDT [14] lead to a performance gain by a minimal margin on the UCF101 dataset. We can explain the decrease in prediction rate by fact that this dataset contains action classes with cluttered backgrounds and illumination changes, and TSN is pre-trained on the large-scale ImageNet dataset, which provides large scale size and diversity. Our approach is based on C3D, which is pre-trained on the UCF101 dataset. However, our introduced method outperformed the 3D conv—iDT by 0.9% and the TSN method by 0.7% on the HDMB51 dataset, and showed the highest recognition rate on small-scale datasets. A possible reason for this higher recognition accuracy is that our model is based on a hybrid deep learning model, and the introduction of LSTM temporally works well by capturing the long-term dependencies and boosting the recognition accuracy for complex action categories in the HDMB51 dataset. We can conclude that a combination of LSTM with a 3D convolutional network for the spatiotemporal stream achieves better results and obtains recognition rates of 92.9% and 70.1%

on UCF101 and HDMB51 datasets respectively. This shows that there is a degree of complimentary between LSTM and convolutional neural network.

## 5. Conclusions

In this paper, we proposed an action recognition framework by utilizing frame-level deep features of the 3D-CNN and processing it through LSTM. First, we introduced a 3Dconv-based model MMN and its iterative training method to integrate the discriminative information of a video into motion maps. Three-dimensional convolutional components extract compact and efficient spatiotemporal features from the input video in the form of feature maps. Moreover, we design a linear weighted fusion method to effectively fuse spatial and temporal feature maps. Finally, we adopt LSTM encoder/decoder to obtain video level representations to conduct video classification. According to the experimental results, our model takes the complementary information contained in multiple features (both spatial and motion features). It is also proof that the motion maps generated by our model intuitively integrate the dynamic information in an efficient manner, and that they retain more discriminative aspects. Moreover, our fusion method makes the features more detailed and specific. To verify the effectiveness of our framework, extensive experiments have been carried out on benchmark datasets, and the obtained results showed that our approach achieves promising performance.

**Author Contributions:** For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, S.A. and J.W.; Methodology, S.A. and Z.F.; Software, T.U.H.; Validation, S.A., T.U.H. and J.W.; Formal Analysis, S.A. and T.U.H.; Investigation, S.A. and T.U.H.; Resources, J.W.; Data Curation, T.U.H.; Writing-Original Draft Preparation, S.A.; Writing-Review & Editing, S.A. and Z.F.; Visualization, S.A.; Supervision, J.W.; Project Administration, J.W. and Z.F.

**Acknowledgments:** The research was supported by Research Institute of Communication Technology (RICT) in Beijing Institute of Technology.

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

## References

1. Poppe, R. A survey on vision-based human action recognition. *Image Vis. Comput.* **2010**, *28*, 976–990. [CrossRef]
2. Laptev, I. On space-time interest points. *Int. J. Comput. Vis.* **2005**, *64*, 107–123. [CrossRef]
3. Willems, G.; Tuytelaars, T.; Gool, L. An efficient dense and scale-invariant spatio-temporal interest point detector. In Proceedings of the 10th European Conference on Computer Vision, Marseille, France, 12–18 October 2008; pp. 650–663.
4. Yeffet, Y.; Wolf, L. Local trinary patterns for human action recognition. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 492–497.
5. Dollr, P.; Rabaud, V.; Cottrell, G.; Belongie, S. Behavior recognition via sparse spatio-temporal features. In Proceedings of the 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Beijing, China, 15–16 October 2005; pp. 65–72.
6. Scovanner, P.; Ali, S.; Shah, M. A 3-dimensional SIFT descriptor and its application to action recognition. In Proceedings of the 15th ACM international conference on Multimedia, Augsburg, Germany, 25–29 September 2007; pp. 357–360.
7. Matikanen, P.; Hebert, M.; Sukthankar, R. Trajectons: Action recognition through the motion analysis of tracked features. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Kyoto, Japan, 27 September–4 October 2009; pp. 514–521.
8. Sun, J.; Wu, X.; Yan, S.; Cheong, L.; Chua, T.S.; Li, J. Hierarchical spatio-temporal context modeling for action recognition. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 2004–2011.
9. Wang, H.; Klser, A.; Schmid, C.; Liu, C.L. Dense trajectories and motion boundary descriptors for action recognition. *Int. J. Comput. Visi.* **2013**, *103*, 60–79. [CrossRef]

10. Wang, H.; Schmid, C. Action recognition with improved trajectories. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 3551–3558.
11. Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; Gould, S. Dynamic image networks for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3034–3042.
12. Chen, C.; Liu, K.; Kehtarnavaz, N. Real-time human action recognition based on depth motion maps. *J. Real-Time Image Process.* **2016**, *12*, 155–163. [CrossRef]
13. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning spatiotemporal features with 3d convolutional networks. *arXiv* **2015**, arXiv:1412.0767.
14. Ji, S.; Xu, W.; Yang, M.; Yu, K. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 221–231. [CrossRef] [PubMed]
15. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Li, F.F. Large-scale video classification with convolutional neural networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
16. Simonyan, K.; Zisserman, A. Two-stream convolutional networks for action recognition in videos. *arXiv* **2014**, arXiv:1406.2199.
17. Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. *arXiv* **2016**, arXiv:1608.00859.
18. Feichtenhofer, C.; Pinz, A.; Zisserman, A. Convolutional two-stream network fusion for video action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1933–1941.
19. Wang, L.; Qiao, Y.; Tang, X. Action recognition with trajectory-pooled deep-convolutional descriptors. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
20. Lu, X.; Yao, H.; Zhao, S. Action recognition with multi-scale trajectory-pooled 3D convolutional descriptors. *Trans. Multimedia Tools Appl.* **2017**, 1–17. [CrossRef]
21. Taylor, G.; Fergus, R.; LeCun, Y.; Bregler, C. Convolutional learning of spatiotemporal features. In Proceedings of the 11th European Conference on Computer Vision, Heraklion, Crete, Greece, 5–11 September 2010; pp. 140–153.
22. Fernando, B.; Gavves, E.; Oramas, J.M.; Ghodrati, A.; Tuytelaars, T. Modeling video evolution for action recognition. In Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 5378–5387.
23. Perronnin, F.; S´anchez, J.; Mensink, T. Improving the fisher kernel for large-scale image classification. In Proceedings of the 11th European Conference on Computer Vision, Heraklion, Crete, Greece, 5–11 September 2010; pp. 143–156.
24. Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent neural network regularization. *arXiv* **2014**, arXiv:1409.2329.
25. Donahue, J.; Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 677–691. [CrossRef] [PubMed]
26. Veeriah, V.; Zhuang, N.; Qi, G.J. Differential recurrent neural networks for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Santiago, Chile, 7–13 December 2015; pp. 4041–4049.
27. Yue-Hei, J.; Hausknecht, M.; Vijayanarasimhan, S. Beyond short snippets: Deep networks for video classification. In Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4694–4702.
28. Wu, Z.; Wang, X.; Jiang, Y. Modelling spatial-temporal clues in a hybrid deep learning framework for video classification. In Proceedings of the 23rd ACM international conference on Multimedia, Brisbane, Australia, 26–30 October 2015; pp. 461–470.
29. Ji-Hae, K.; Gwang-soo, H.; Byung-Gyu, K.; Debi, D. deepGesture: Deep learning-based gesture recognition scheme using motion sensors. *Displays* **2018**, *55*, 38–45.
30. Srivastava, N.; Mansimov, E.; Salakhutdinov, R. Unsupervised Learning of Video Representations using LSTMs. *arXiv* **2015**, arXiv:1502.04681.

31. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
32. Soomro, K.; Zamir, A.R.; Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv* **2012**, arXiv:1212.0402.
33. Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. Hmdb: A large video database for human motion recognition. In Proceedings of the IEEE 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2556–2563.
34. Rodriguez, M.D.; Ahmed, J.; Shah, M. Action MACH a spatiotemporal maximum average correlation height filter for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern (CVPR), Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
35. Jiang, Y.G.; Liu, J.; Zamir, R.; Laptev, I.; Piccardi, M.; Shah, M.; Sukthankar, R. THUMOS challenge: Action Recognition with a Large Number of Classes. The First International Workshop on Action Recognition with a Large Number of Classes, in Conjunction with ICCV'13, Sydney, Australia. 2013. Available online: http://crcv.ucf.edu/ICCV13-Action-Workshop/ (accessed on 28 January 2019).
36. Murthy, V.R.; Goecke, R. Ordered trajectories for large scale human action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 25–27 June 2013; pp. 412–419.
37. Ni, B.; Moulin, P.; Yang, X. Motion part regularization: Improving action recognition via trajectory selection. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3698–3706.
38. Wang, L.; Qiao, Y.; Tang, X. Mofap: A multi-level representation for action recognition. *Int. J. Comput. Vis.* **2016**, *119*, 119–254. [CrossRef]
39. Seo, J.; Kim, H.; Ro, Y.M. Effective and efficient human action recognition using dynamic frame skipping and trajectory rejection. *J. Image Vis. Comput.* **2017**, *58*, 76–85. [CrossRef]
40. Sun, L.; Jia, K.; Shi, B.E. Human action recognition using factorized spatio-temporal convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4597–4605.
41. Zhang, B.; Wang, L.; Wang, Z.Y. Real-time action recognition with enhanced motion vector CNNs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2718–2726.
42. Wang, J.; Wang, W.; Wang, R. Deep alternative neural network: Exploring contexts as early as possible for action recognition. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 4–9 December 2016; pp. 811–819.
43. Varol, G.; Laptev, I.; Schmid, C. Long-term temporal convolutions for action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 1510–1517. [CrossRef] [PubMed]
44. Li, Z.; Gavves, E.; Jain, M. VideoLSTM convolves, attends and flows for action recognition. *Comput. Vis. Image Underst.* **2016**, *166*, 41–50. [CrossRef]
45. Wang, X.; Gao, L.; Wang, P.; Liu, X. Two-stream 3D convNet Fusion for Action Recognition in Videos with Arbitrary Size and Length. *IEEE Trans. Multimedia* **2017**, *20*, 1–11.
46. Yu, S.; Cheng, Y.; Xie, L. Fully convolutional networks for action recognition. *Inst. Eng. Technol. Comput. Vis.* **2017**, *11*, 744–749. [CrossRef]
47. Zhu, Y.; Lan, Z.; Newsam, S. Hidden two-stream convolutional networks for action recognition. *arXiv* **2017**, arXiv:1704.00389.
48. Yeung, S.; Russakovsky, O.; Jin, N. Every moment counts: Dense detailed labelling of actions in complex videos. *Int. J. Comput. Vis.* **2018**, *126*, 375–389. [CrossRef]
49. Wang, X.; Gao, L.; Song, J.; Shen, H. Beyond frame-level CNN: Saliency-aware 3-D CNN with LSTM for video action recognition. *IEEE Signal Process. Lett.* **2017**, *24*, 510–514. [CrossRef]

*future internet*

*Article*

# Forward-Looking Element Recognition Based on the LSTM-CRF Model with the Integrity Algorithm

**Dong Xu, Ruping Ge * and Zhihua Niu**

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; dxu@shu.edu.cn (D.X.); ZhNiu@shu.edu.cn (Z.N.)
* Correspondence: geruping@shu.edu.cn; Tel.: +86-188-1760-8630

Received: 31 October 2018; Accepted: 8 January 2019; Published: 14 January 2019

**Abstract:** A state-of-the-art entity recognition system relies on deep learning under data-driven conditions. In this paper, we combine deep learning with linguistic features and propose the long short-term memory-conditional random field model (LSTM-CRF model) with the integrity algorithm. This approach is primarily based on the use of part-of-speech (POS) syntactic rules to correct the boundaries of LSTM-CRF model annotations and improve its performance by raising the integrity of the elements. The method incorporates the advantages of the data-driven method and dependency syntax, and improves the precision rate of the elements without losing recall rate. Experiments show that the integrity algorithm is not only easy to combine with the other neural network model, but the overall effect is better than several advanced methods. In addition, we conducted cross-domain experiments based on a multi-industry corpus in the financial field. The results indicate that the method can be applied to other industries.

**Keywords:** LSTM-CRF model; elements recognition; linguistic features; POS syntactic rules

## 1. Introduction

In recent years, thanks to the broad application of natural language processing (NLP) technology, financial information processing capabilities have been unprecedentedly improved. For example, studies have explored the effects of financial events on stock price predictions [1,2], used firm reports to predict corporate performance [3], performed financial text sentiment analysis [4], and identified and extracted financial concepts (financial named entities, FNE) [5]. Notably, firm research reports have always been an important source of financial information. This information can be used to analyze the recent situation of a company from a professional perspective, make predictive assessment regarding the economic patterns and trends of the company in upcoming years, and provide professional investment advice. Previous studies [6] have also noted that compared to individual subjective texts, such as stocks and forums, the firm research reports are more realistic, and can provide a reliable source for financial text sentiment calculations and financial early warning decisions.

In 1980, the American Stock Exchange required all listed companies to include a Management Discussion and Analysis (MD&A) section in their annual reports. MD&A focuses on the disclosure of forward-looking statements (FLSs) that may have a significant impact on the company. An FLS is an assessment and expectation of the future development trends and prospects of the company, and such information is of great significance to venture investors and other stakeholders. In recent years, many scholars have analyzed and thoroughly explored FLSs. For example, Feng Li [7] used a naive Bayes classifier to explore the correlation between the information contained in FLSs in annual reports and economic factors. In article, our objective is to achieve the fine-scale recognition of forward-looking sentences in research reports; for example, “原料药业务贡献的净利润将显著增厚” (The net profit contributed by the drug substance business will increase significantly). The element triple <Entity, Attribute, Attribute value> corresponds to <原料药业务 (drug substance business), 净利润 (net profit), 显著增厚 (significant increase)> in the sentence. This basic task can be applied in high-level applications such as sentiment analysis, investment decision making, and stock forecasting.

Difficulties encountered in our work mainly include the following points. First, unlike other fields elements, finance is a relatively open field. Entities or attributes are very broad and difficult to identify. Second, new entities are constantly emerging. It is difficult to develop simple template rules that apply to all situations. Third, in the financial corpus, attributes, and attribute value elements appear mainly in the form of compound words or clauses. Therefore, the integrity of the recognition of the elements must also be considered. Deep learning can be used to overcome the two problems of open fields and new entities. In particular, deep learning is a data-driven approach that automatically identifies elements and builds models based on learning appropriate to the field. However, natural language itself is a highly symbolic and complex discrete rule, it is a collection of conventions and domain knowledge associated with the process of human evolution. In a purely data-driven approach, the language may lose its original meaning. Therefore, we combine deep learning with linguistic features and propose the LSTM-CRF model with the integrity algorithm, mainly to improve the recognition effect by correcting the boundaries of LSTM-CRF model annotations. This method combines the advantages of data-driven methods and dependency grammar to improve the accuracy and recall of elements.

The structure of this paper is as follows. Section 2 introduces the status of element recognition research. Section 3 describes forward-looking element recognition based on the LSTM-CRF model with the integrity algorithm. Section 4 details the experimental steps and the analysis of the experimental results. The Section 5 is the conclusion of the paper and outlooks for future work.

## 2. Related Work

The research objective of this article is similar to that of Feldman et al. [8] The goal is to obtain various elements of specified type at the sentence level, including entities, attributes, and attribute values. The existing element recognition methods can be largely divided into the following three categories.

### *2.1. Rule/Dictionary Methods*

Feldman et al. [8] used regular expressions to extract product names and attributes in the construction of running shoes based on an automotive product brand dictionary and attribute dictionary. Another study [9] used association rules to mine frequent nouns or noun phrases in product reviews as features of recognition. Moreover, Reference [10] proposed a rule-based recognition strategy based on the syntactic relationships among opinion words and the target, and they expanded the initial opinion lexicon and extracted the target using a propagation algorithm.

### *2.2. Machine Learning*

Hai et al. [11] extracted a maximum entropy model for ontological event elements to obtain the five tuples in comparative sentences, namely, the comparison subject, comparison object, comparison attribute, comparison word, and evaluation word. The CRF model does not rely on independence assumptions; therefore, it can be used to fuse complex non-local features and better capture potential relationships among states. Therefore, it is widely used in entity recognition tasks. A previous study [12] proposed the combination of domain knowledge and vocabulary information using the CRF model to identify product features. Finkel [13] proposed an automatic tagging model that considers the characteristics of words, including suffixes, part-of-speech sequences and the morphology of words. Choi [14] used the CRF model, fusion words, part-of-speech features, opinion lexicon features, and dependency tree features to assess recognition from a specific viewpoint.

### *2.3. Deep Learning Methods*

With the development of word-distributed representation, deep learning is a powerful tool for sequence modeling. Typically, a word is expressed by word embedding and then input into the neural network model. In this process, the hidden layer of the multilayer structure is used to extract features and predict the label. Collobert et al. [15] combined a convolutional neural network (CNN) with a CRF to achieve better results for named entity recognition tasks. Huang et al. [16] presented a bidirectional LSTM-CRF model (Bi-LSTM-CRF model) for NLP benchmark sequence tagging data. The model yielded an F-value of 88.83% for the CONLL2003 corpus. Limsopatham et al. [17] used the Bi-LSTM framework to automatically learn orthographic features and investigate the named entity recognition problem. The proposed approach performed excellently in “segmentation and categorization” and “segmentation only” subtasks. Lample et al. [18] constructed a LSTM-CRF model using word representations and character-based representations of captured morphological and orthogonal information, and the model performed well in named entity recognition (NER) tasks in four languages.

In the above studies, the rule methods are effective, but they require manual construction and domain knowledge. Machine learning models require large numbers of corpus-based and manual features, and the recall rate is not high. Although deep learning models provide satisfactory recognition effects, they ignore the meaning of the language itself and cannot be further improved. Analyses of financial domain corpora have shown that these methods cannot be directly applied to the financial field. Notably, the openness and integrity of elements must be considered.

In recent years, dependency grammar has performed well in identifying tasks. Popescu [19] and others used the rules of dependency syntax to recognize emotional words. Bloom et al. [20] used the rules of syntactic dependency to formulate rules and identify evaluation objects and evaluation word pairs. Somprasertsri et al. [21] used dependency syntax to build evaluation objects and candidate sets of evaluation words and obtain evaluation objects and evaluation words. Then, the candidate set was reselected using a maximum entropy model to obtain the final evaluation object and evaluation word pair. Therefore, we can rely on the syntax integrity algorithm to correct the element boundaries and combine it with the LSTM-CRF model to propose the LSTM-CRF model with the integrity algorithm. This method has the following three advantages over the conventional method:

1. The dependency syntax can capture the long-range collocation and modification relations between the internal components of a sentence. This feature can be used to solve the problem of integrity;
2. Open-element and new entity issues can be solved with the superior recognition performance of the LSTM-CRF model;
3. The proposed model meets the requirement of simultaneously identifying different types of elements.

## **3. LSTM-CRF Model with the Integrity Algorithm**

The proposed element recognition method based on the LSTM-CRF model is mainly composed of the LSTM-CRF model and integrity algorithm. This section starts by giving an overview of our model. Next, we will describe these two parts of the proposed method in detail.

### *3.1. Model Overview*

The model architecture is showed in Figure 1. It first uses the LSTM-CRF model to obtain the tag sequence ($LS_1$, $LS_2$ ... $LS_n$) under the influence of a data drive. The input is a word vector corresponding to a word, and the word vector can be pre-trained or trained together in the model. The output is the label for each word. This process can only get a rough range of each element. The most important idea of the LSTM-CRF model is to add the CRF model as the decoding layer of the model based on Bi-LSTM and consider the rationality between the prediction results. To further obtain accurate boundaries, the integrity algorithm formed by the POS syntactic rules is used to continuously correct the range of element recognition to obtain the final tag sequence ($L_1$, $L_2$ ... $L_n$).

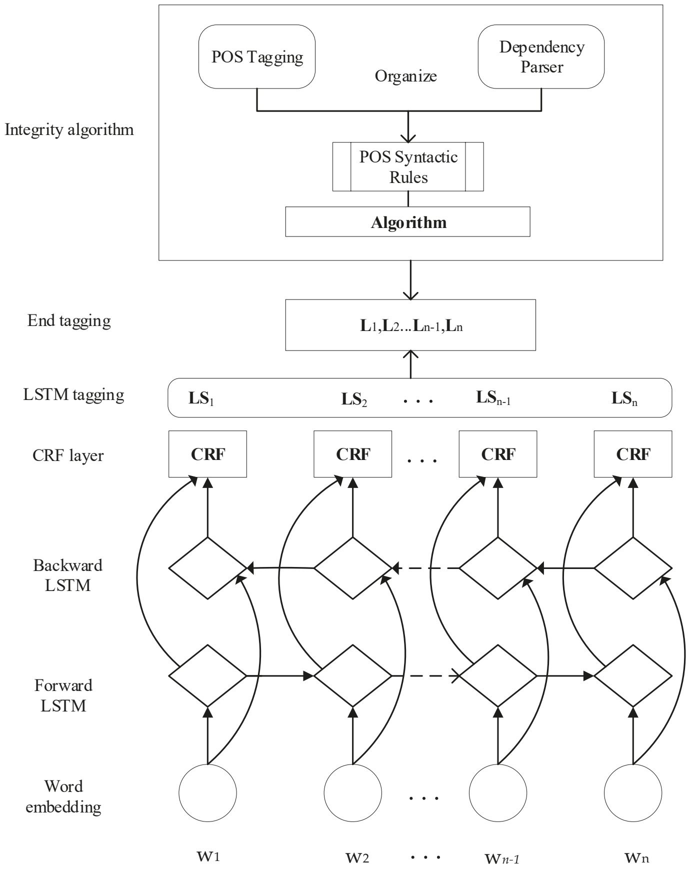


**Figure 1.** The main architecture of our model.

## 3.2. LSTM-CRF Model

LSTM networks are a specific form of recurrent neural network (RNN) proposed by Sepp Hochreiter, Jurgen Schmidhuber, and colleagues [22]. Unlike the general RNN model, conventional neurons are replaced with memory cells, each of which consists of input gates, forget gates, and output gates. This approach not only allows long-term dependencies to be identified but also avoids the problem of gradient disappearance or gradient expansion. In this article, the LSTM-CRF model is like that proposed by Huang et al. [16]. The model can be formulated as follows:

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_{xi}\, \boldsymbol{x}_t + \boldsymbol{W}_{hi}\, \boldsymbol{h}_{t-1} + \boldsymbol{W}_{ci}\, \boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right) \tag{1}$$

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_{xf}\, \boldsymbol{x}_t + \boldsymbol{W}_{hf}\, \boldsymbol{h}_{t-1} + \boldsymbol{W}_{cf}\, \boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right) \tag{2}$$

$$\widetilde{c} = tanh\left(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) \tag{3}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \widetilde{c} \tag{4}$$

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_{xo}\, \boldsymbol{x}_t + \boldsymbol{W}_{ho}\, \boldsymbol{h}_{t-1} + \boldsymbol{W}_{co}\, \boldsymbol{c}_t + \boldsymbol{b}_o\right) \tag{5}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t) \tag{6}$$

where *f*, *i*, and *o* represent the forget, input, and output gates, respectively; $c_{t-1}$ represents the state of the cell at time $t-1$; $c_t$ represents the state of the cell at time $t$. $h_t$ represents the output of the current state; $h_{t-1}$ represents the output of the unit at the previous moment; σ is the logistic sigmoid function; W and b represent the weight and bias, respectively; and ⊙ is the element-wise product.

The CRF model is an undirected graph model that was proposed by Lafferty et al. [23] in 2001. The model has obvious advantages in labeling and segmenting serialized data. It directly models the conditional distribution of data and can effectively avoid the problem of mark biasing experienced by other discriminant models. Moreover, the CRF model is different from other production models because it does not rely on the independence assumption, which allows it to fuse a variety of complex, non-local features and better capture the potential relationships among states.

The goal of the CRF model is to calculate the conditional probability distribution of the optimal output sequence (predictive sequence) given the input sequence (observed sequence), i.e., $P(Y|X)$. For example, assume that the random variable $X = (x_1, x_2, \ldots, x_{n-1}, x_n)$ is the input sequence and the random variable $Y = (y_1, y_2, \ldots, y_{n-1}, y_n)$ is the output sequence. Then, under the condition that the observation sequence is X, the conditional probability distribution of the prediction sequence Y is as shown in Formulas (7) and (8):

$$P(X|Y) = \frac{1}{Z(X)} \exp(\sum_k \lambda_k \alpha_k (y_{i-1}, y_i, x, i) + \sum_k \mu_k \beta_k (y_i, x, i)) \tag{7}$$

$$Z(X) = \sum_y \exp(\sum_k \lambda_k \alpha_k (y_{i-1}, y_i, x, i) + \sum_k \mu_k \beta_k (y_i, x, i)) \tag{8}$$

where $\alpha_k$ and $\beta_k$ are binary functions (0 or 1); $\alpha_k$ is a transfer eigenfunction that reflects the correlation between adjacent marker variables at two positions, $i-1$ and $i$, and the effect of observation sequences on marker variables; $\beta_k$ is a state eigenfunction that represents the effect of a node at observation sequence position i on the marker variable; $\lambda_k$ and $\mu_k$ are parameters corresponding to the characteristic functions $\alpha_k$ and $\beta_k$, respectively; and *Z(X)* is the normalization factor.

The LSTM-CRF model is widely used in the NER task. The powerful data fitting ability of the LSTM model can be utilized, and sequence labeling can be directly implemented by the CRF. The labeling of each current word is related to the labeling result at the previous moment. The LSTM-CRF model consists of a word embedding layer, a bidirectional LSTM layer, and a CRF layer. The first word embedding layer uses pretraining to map each word $w_i$ of a sentence into a *d*-dimensional word vector. The LSTM-CRF model architecture is shown in Figure 2.

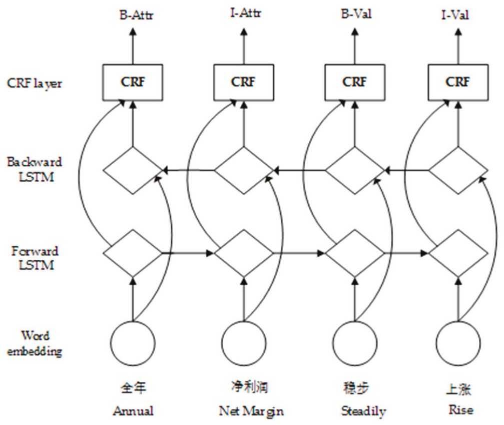


**Figure 2.** LSTM-CRF model architecture.

The word vector is input into the bidirectional LSTM layer, and the hidden state of the forward LSTM output and the hidden state of the backward LSTM output are obtained. The hidden state sequence output by each position is spliced by position to obtain a complete hidden state sequence $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$.

Next, P is regarded as the matrix of scores outputted by the LSTM network of size $n \times k$, where $k$ indicates the category of the tags. $\mathrm{P}_{i,\,j}$ corresponds to the score of the j-th tag of the i-th word in a sentence. Given an input sentence $X = (x_1, x_2, \ldots, x_n)$ and a sequence of predictions, $y = (y_1, y_2, \ldots, y_n)$, the associated score is defined as follows:

$$score(X, y) = \sum_{i=1}^{n} P_{i,yi} + \sum_{i=1}^{n} A_{yi-1,yi} \qquad s.t.P \in R^{n\times k}, A \in R^{(k+2)\times(k+2)} \tag{9}$$

where $A_{i,j}$ is the transfer score from the $i$th tag to the $j$th tag. The start and end tags are added to the set of possible tags, and these tags, namely, $y_0$ and $y_n$, denote the start and end of a sentence. Thus, $A$ is a square matrix of size $k + 2$. After applying a softmax function to all possible tag sequences, the probability of the sequence $y$ is as follows:

$$P(y|X) = \frac{\exp(score(X, y))}{\sum_{y\prime} \exp(score(X, y\prime))} \tag{10}$$

The model is trained by maximizing the log likelihood function:

$$\log P(y^x|X) = score(X, y^x) - \log(\sum_{y\prime} score(X, y\prime)) \tag{11}$$

Finally, the Viterbi algorithm is used to solve the optimal path:

$$y^* = \arg\max_{y\prime} score(X, y\prime) \tag{12}$$

## 3.3. Integrity Algorithm

### 3.3.1. Dependency Syntax

To improve the integrity of element recognition, the dependency syntax relationship is introduced, which can capture the long-range collocation and modification relationships. This relationship is combined with the part-of-speech (POS) rules to filter out useless rules that occasionally occur. Then, the POS syntactic rules are organized to identify the complete structure of the elements. The LTP dependency syntax analysis based on the cloud platform (https://www.ltp-cloud.com/), each dependency is composed of core words and modifiers. The dependency relationship between two words is connected by a dependency arc, and the specific relationship between collocations is indicated by the marks on the dependency arc. Tags are shown in Table 1.

**Table 1.** Tags of LTP-cloud Platform Dependency Syntax Analysis.

| Tags | Dependency Relations | Tags | Dependency Relations |
|---|---|---|---|
| SBV | Subject-verb | CMP | Complement |
| VOB | Verb object | COO | Coordinate |
| FOB | Fronting object | POB | Preposition object |
| DBL | Double | LAD | Left adjunct |
| IOB | Indirect object | RAD | Right adjunct |
| ATT | Attribute | IS | Independent structure |
| ADV | Adverbial | HED | Head |

By reviewing 1050 financial reports, we found that attributes are often composed of compound words, such as "利润增速水平" (profit growth rate). Even attribute values are clause structures, such as "增长接近100%左右" (the growth is close to 100%). Most of the attribute values in the reports appear in the form of objects, and there are also ATT and COO dependencies among the constituent units. Similarly, some fixed dependencies exist between the constituent units of attributes, such as in the following forward-looking sentence: "预计公司的收入和毛利率将继续呈现高速增长的趋势" (It is expected that the company's revenue and gross profit margin will continue to show a trend of rapid growth). The results of the dependency syntactic analysis are shown in Figure 3.

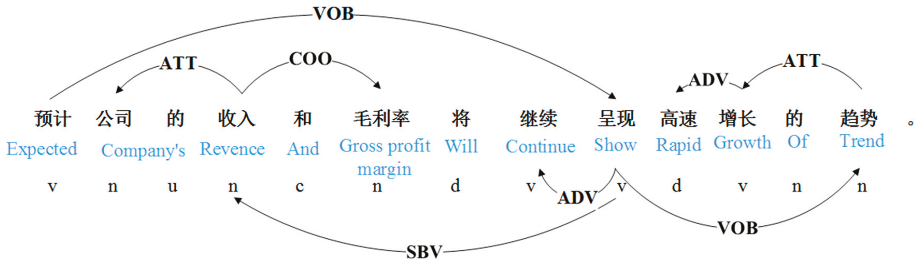


**Figure 3.** Dependency analysis results.

The attributes of this forward-looking sentence are "收入" (revenue) and "毛利率" (gross profit margin), and the syntactic role they play in a sentence is providing the subject for the verb "呈现" (show), namely, there is an SBV relationship. There is also a COO dependency between "收入" (revenue) and "毛利率" (gross profit margin). The attribute value element of the sentence is "呈现高速增长的趋势" (show a trend of rapid growth), which is part of the subordinate clause, where "呈现" (show) is the object of "预计" (expected), namely, a VOB relationship. Within the attribute value, there is an ADV relationship between "高速" (rapid) and "增长" (growth), a VOB relationship between "呈现" (show) and "趋势" (trend), and an ATT dependency relationship between "增长" (growth) and "趋势" (trend).

### 3.3.2. POS Syntactic Rules

As shown in Tables 2 and 3, certain POS syntactic rules are obtained using the context of attributes and attribute values via a corpus analysis. $\text{Rule}_{\text{id}}$ represents the encoding of rules; description represents the specific content of the rule; and output represents the recognized elements. where Ds is the dependency relation between word $W_i$ and $W_j$ ($W_k$) and Ds contains {VOB, ATT, ADV, COO, CMP, … }, where $W_j$ ($W_k$) is the first or last n words of $W_i$. Therefore, the dependency relation is based on the context of $W_i$. POS encompasses the part-of-speech information for $W_i$. Con ($W_i$~$W_j$) represents the connection from $W_i$ to $W_j$. The arrows represent the dependency relations. For example, $W_i \rightarrow Ds \rightarrow W_j$ denotes that Wi depends on $W_j$ through a syntactic relation Ds, where $W_i$ is the father node of W. $R1_i$ indicates that an attribute (Attr) or attribute value (Val) is identified using only one dependency relation, and $R2_i$ indicates that there is more than one type of dependency relationship.

**Table 2.** POS syntactic rules based on the contextual rules of attributes.

| $\text{Rule}_{\text{id}}$ | Description | Output |
|---|---|---|
| $R1_1$ | $W_i \rightarrow D_S \rightarrow W_j$ s.t. $W_j \in \{\text{context}(W_i)\}$, DS = ATT, POS $(W_j, W_i) = \{(n, n), (v, n), (nl, n)\}$ | Con $(W_j \sim W_i)$ |
| $R1_2$ | $W_i \rightarrow D_S \rightarrow W_j$ s.t. $W_j \in \{\text{context}(W_i)\}$, $D_S$ = COO, POS $(W_i, W_j) = (n, n)$ | Con $(W_i \sim W_j)$ |
| $R2_1$ | $W_i \rightarrow D_{S1} \rightarrow W_j$ and $W_i \rightarrow D_{S2} \rightarrow W_k$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = ATT, $D_{S2}$ = ATT, POS $(W_k, W_j, W_i) = \{(n, v, n), (n, n, n)\}$ | Con $(W_k \sim W_j \sim W_i)$ |
| $R2_2$ | $W_i \rightarrow D_{S1} \rightarrow W_j$ and $W_j \rightarrow D_{S2} \rightarrow W_k$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = ATT, $D_{S2}$ = ATT, POS $(W_j, W_i, W_k) = (n, n, n)$ | Con $(W_j \sim W_i \sim W_k)$ |
| $R2_3$ | $W_i \rightarrow D_{S1} \rightarrow W_j$ and $W_k \rightarrow D_{S2} \rightarrow W_i$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = ADV, $D_{S2}$ = ATT, POS $(W_j, W_i, W_k) = (a, v, n)$ | Con $(W_j \sim W_i \sim W_k)$ |

**Table 3.** POS syntactic rules based on the contextual rules of attributes values.

| $\text{Rule}_{\text{id}}$ | Description | Output |
|---|---|---|
| $R1_1$ | $W_i \rightarrow D_S \rightarrow W_j$ s.t. $W_j \in \{\text{context}(W_i)\}$, $D_S$ = VOB, POS $(W_i, W_j) = \{(v, v), (v, m), (v, n)\}$ | Con $(W_i \sim W_j)$ |
| $R1_2$ | $W_i \rightarrow D_S \rightarrow W_j$ s.t. $W_j \in \{\text{context}(W_i)\}$, $D_S$ = ADV, POS $(W_j, W_i) = \{(d, v), (a, v), (d, v), (d, a)\}$ | Con $(W_j \sim W_i)$ |
| $R1_3$ | $W_i \rightarrow D_S \rightarrow W_j$ s.t. $W_j \in \{\text{context}(W_i)\}$, $D_S$ = ATT, POS $(W_i, W_j) = \{(a, n), (m, q), (m, m), (b, n)\}$ | Con $(W_i \sim W_j)$ |
| $R2_1$ | $W_i \rightarrow D_{S1} \rightarrow W_j \rightarrow D_{S2} \rightarrow W_k$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = VOB, $D_{S2}$ = ADV, POS $(W_i, W_k, W_j) = (v, d, v)$ | Con $(W_i \sim W_k \sim W_j)$ |
| $R2_2$ | $W_i \rightarrow D_{S1} \rightarrow W_j$ and $W_j \rightarrow D_{S2} \rightarrow W_k$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = VOB, $D_{S2}$ = RAD, POS $(W_i, W_j, W_k) = (v, m, m)$ | Con $(W_i \sim W_j \sim W_k)$ |
| $R2_3$ | $W_i \rightarrow D_{S1} \rightarrow W_j$ and $W_j \rightarrow D_{S2} \rightarrow W_k$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = VOB, $D_{S2}$ = ATT, POS $(W_i, W_k, W_j) = (v, b, n)$ | Con $(W_i \sim W_k \sim W_j)$ |
| $R2_4$ | $W_i \rightarrow D_{S1} \rightarrow W_j$ and $W_i \rightarrow D_{S2} \rightarrow W_k$ s.t. $W_j, W_k \in \{\text{context}(W_i)\}$, $D_{S1}$ = ADV, $D_{S2}$ = CMP, POS $(W_j, W_i, W_k) = (d, v, a)$ | Con $(W_j \sim W_i \sim W_k)$ |

### 3.3.3. Algorithm

Although a more complete integrity structure of elements can be obtained through the POS syntactic rules, the number of rules remains limited. The LSTM-CRF model can greatly improve the element recall rate, although the integrity is not guaranteed with this approach. Therefore, we can combine the advantages of both models and propose the LSTM-CRF model with the integrity algorithm. The method incorporates the advantages of the data-driven method and dependency syntax to raise the integrity and improve the accuracy of elements without losing the element recall rate.

The inputs of the LSTM-CRF model with the integrity algorithm include $\mathbf{LS}_{\text{tag}}$, $\mathbf{R}_{\text{tag}}$, and the following user specified parameters:

1. $\mathbf{LS}_{\text{tag}}$, the result set labeled by the LSTM-CRF model, including {B-Attr ("B-Attr" means the beginning of the attribute element), I-Attr ("I-Attr" means the reminder of attribute element), O ("O" means not belonging to any type of element)};
2. $\mathbf{R}_{\text{tag}}$, the result set labeled based on the POS syntactic rules, including {B-Attr, I-Attr, O};
3. **L**, the tags result set;
4. $\mathbf{R}_{\text{tag\_begin}}$, the subscript of the starting position of each attribute (B-Attr);
5. $\mathbf{R}_{\text{tag\_end}}$, the subscript of the ending position of each attribute (I-Attr);
6. flag, as a flag, its value is true or false;
7. $i$, as a variable to identify the position of each tag.

The main idea of the algorithm is to utilize the tags generated by the POS syntactic rules to correct the tags generated by the LSTM-CRF model. The algorithm can be divided into three steps. First, $\mathbf{R}_{\text{tag}}$ is traversed to find B-Attr, the current position is set as the superscript, and look for the subscript is obtained from the current label position [$\mathbf{R}_{\text{tag\_begin}}$: $\mathbf{R}_{\text{tag\_end}}$]. Second, $\mathbf{LS}_{\text{tag}}$ [$\mathbf{R}_{\text{tag\_begin}}$: $\mathrm{R}_{\text{tag\_end}}$] is traversed to determine whether the label of the interval only includes O and the label; if it does, then $\mathbf{LS}_{\text{tag}}$ is covered with $\mathbf{R}_{\text{tag}}$ in the interval, and otherwise, it will not be replaced. Third, the next set of annotations after $\mathbf{R}_{\text{tag\_end}}$ is searched until $\mathbf{R}_{\text{tag}}$ is traversed. By correcting the recognition strategy, the integrity of the elements will be guaranteed as much as possible. The LSTM-CRF model with integrity algorithm is shown in Algorithm 1.

**Algorithm 1.** LSTM-CRF model with integrity algorithm

```
Input: $\mathbf{LS}_{tag}$, $\mathbf{R}_{tag}$, flag, $i$
Output: $\mathbf{L}$
1: Take the result of attribute tagging as an example
2: for ($i$ = 0; $i$ < $\mathbf{R}_{tag}$.length; $i$++)
3:     set $\mathbf{R}_{tag\_begin}$ = 0, $\mathbf{R}_{tag\_end}$ = 0
4:     if ($\mathbf{R}_{tag}$ [$i$]! = O)
5:        $\mathbf{R}_{tag\_begin}$ = $i$;
6:     while ($\mathbf{R}_{tag}$ [$i$]! = O)
7:          $i$++; $\mathbf{R}_{tag\_end}$ = $i$;
8:        flag = true;
9:        for ls $\in$ $\mathbf{LS}_{tag}$ [$\mathbf{R}_{tag\_begin}$: $\mathbf{R}_{tag\_end}$]
10:         if ls! = O or ls! = $\mathbf{R}_{tag}$ [$i$]
11:            flag = false;
12:         if (flag)
13:            $\mathbf{LS}_{tag}$ [$\mathbf{R}_{tag\_begin}$: $\mathbf{R}_{tag\_end}$] = $\mathbf{R}_{tag}$ [$\mathbf{R}_{tag\_begin}$: $\mathbf{R}_{tag\_end}$]
14:            $\mathbf{L} \leftarrow \mathbf{LS}_{tag}$
15: return $\mathbf{L}$
```

## 4. Experiment

This section evaluates the performance of our proposed method through experiments and a results analysis. The first part describes the source of the data, the second part introduces the evaluation indicators, and the third part describes the training settings of the LSTM-CRF model. Finally, two sets of experiments show that the LSTM-CRF model with the integrity algorithm provides better results compared with the LSTM-CRF model and POS syntactic rules and our proposed method has good domain independence.

### *4.1. Data Description*

The object of this article's research is to determine whether elements in firm research reports can be recognized at the sentence level. However, Chinese firm reports tagging their corpus are not publicly available. Therefore, we have constructed a forward-looking information corpus of Chinese firm research reports. Financial websites are crawled to get the experimental data, the source data are extracted in html, and the crawled pages are denoised. The size of the corpus is shown in Table 4.

To standardize the data set, reports from different companies in different fields are used and students from financial fields are invited as annotators, with three students used to annotate the same data. The results are determined by the principle of the minority obeying the majority.

**Table 4.** Corpus size statistics, where En, Attr, and Val represents entity, attribute, and value, respectively.

| Fields | Articles | FLSs | En | Attr | Val |
|---|---|---|---|---|---|
| Pharmaceutical industry | 600 | 1362 | 1587 | 1887 | 3876 |
| Medical industry | 225 | 594 | 726 | 912 | 1344 |
| Car manufacturer | 225 | 197 | 1026 | 1011 | 1431 |
| Total | 1050 | 2547 | 3339 | 3810 | 6651 |

### *4.2. Evaluation Indicators*

The recognition effectiveness can be evaluated based on the precision (P), recall (R), and F-score (F). To better assess the experimental results, accuracy and coverage evaluations were conducted.

Accuracy evaluations: The recognition results are compared to manual markings and are required to be fully compliant.

Coverage evaluation: This requires that the recognition result partially overlaps with the corpus of the tag. However, due to the different lengths of the constituent elements of the entities, attributes, and attribute values in this paper, the coverage evaluation cannot be unified.

In most cases, entities and attributes are formed by combining two nouns. The standard for coverage evaluation of entities and attributes is a partial overlap between the recognition results and manual markers. Denoted as "利润率增速" (Profit rate growth), the recognition results of "利润率" (Profit rate) and "增速" (growth) are considered to match.

The composition of the attribute value is longer than that of other elements, and partial matching can lead to unclear results. Therefore, the coverage evaluation of attribute values requires the recognition results to contain the manual markers, which is considered a correct match. For example, for the label "同比上升约2.5个百分点" (increased by about 2.5 percentage points year-on-year), "将同比上升约2.5个百分点" (will increase by about 2.5 percentage points year-on-year) is the correct match.

### *4.3. Experimental Settings*

Our experiments are based on the Python language and the TensorFlow platform. We used LTP as the tool for word segmentation POS dependency syntax analysis. Experiments are implemented in a Python 3.6 version on CentOS 7.4 running on a server with a system configuration 16-core Intel Xeon E5 processor (2.10 GHz) with 16 GB RAM.

During the training of the network model, the pre-trained embedding with dimensions of 100 generated by the continuous bag-of-words (CBOW) model is directly used as the input of the LSTM-CRF model and used the Sogou news data (http://www.sogou.com/labs/resource/list_news.php) as the corpus to train the word embedding. To further improve the performance of the model and prevent overfitting, the dropout rate was fixed at 0.5 for all dropout layers in all experiments and the LSTM-CRF model was trained using backpropagation algorithms to optimize the parameters. In addition, the epoch was 20, and the batch size was 50. Stochastic gradient decent (SGD) algorithm was adopted with a learning rate of 0.05 for every epoch based on the training set.

### *4.4. Experimental Results and Analysis*

To obtain reasonable and credible results, we use the corpus of the pharmaceutical industry to conduct five-fold crossover experiments. In this experiment, the experimental data are randomly divided into five subsets, of which four were used as training sets and one was used as the test set. This process was repeated five times.

#### 4.4.1. The Results of the Five-Fold Crossover Experiments

In Table 5, we can see that the LSTM-CRF model performed better overall than the rules method because the LSTM has a strong sequence modeling advantage and the CRF can optimize the entire sequence to make up for the local optimization problem of the LSTM. Therefore, the performance of the LSTM-CRF in the NER task was outstanding, which has been demonstrated by many previous studies. However, compared with the rules method, the precision indicator of the LSTM-CRF model was slightly lower because it is more driven by data and the text is converted to a vector at the time of input, which accelerates the data processing. However, this process also leads to the loss of many features of the language itself. Therefore, in the recognition results, we can find the identified boundaries of long elements, such as Attr and Val, were not sufficiently accurate; thus, the integrity of element recognition needs to be improved. The POS syntactic rules method presents lower recall and slightly higher precision because the rules of manual collection are always limited.

To incorporate the advantages of both, the LSTM-CRF model with the integrity algorithm (our method) is proposed, which can improve the recall rate under the data-driven approach and improve the precision under the syntactic dependence. Table 5 demonstrates that our method works better than the LSTM-CRF model and the POS syntactic rule method.

In Table 5, the coverage evaluation indexes of the LSTM-CRF model, POS syntactic rules, and our model are higher than that of the accuracy evaluation. Moreover, the precision of attribute value of the LSTM-CRF model (P = 85.99) was higher than that of the rules (82.83) under the coverage evaluation, indicating that the positioning of each element by the LSTM-CRF was relatively accurate while the positioning of element boundaries needs further improvement, which is the focus of this paper.

**Table 5.** Average the results of five-fold crossover experiments (%).

| **Elements** | **Model** | **Accurate Evaluation** | | | **Coverage Evaluation** | | |
|---|---|---|---|---|---|---|---|
| | | **P** | **R** | **F** | **P** | **R** | **F** |
| En | LSTM-CRF | 56.38 | 38.53 | 45.75 | 70.28 | 60.43 | 64.96 |
| | Rules | - | - | - | - | - | - |
| | Our method | 64.39 | 45.08 | 52.98 | 77.33 | 65.82 | 71.09 |
| Attr | LSTM-CRF | 71.35 | 58.67 | 64.19 | 81.79 | 70.57 | 75.69 |
| | Rules | 79.72 | 51.57 | 62.47 | 83.26 | 61.88 | 70.65 |
| | Our method | 81.86 | 63.59 | 71.47 | 86.46 | 74.17 | 79.74 |
| Val | LSTM-CRF | 76.81 | 72.61 | 74.62 | 85.99 | 77.31 | 81.38 |
| | Rules | 81.55 | 57.01 | 66.86 | 82.83 | 67.87 | 74.52 |
| | Our method | 86.09 | 73.62 | 79.34 | 90.17 | 81.84 | 85.64 |

From the perspective of the type of element, the indicators suggest that the entity recognition results (F = 45.75) were poor compared to those for the attributes (F = 64.19) and attribute values (F = 74.62), which can be explained as follows: first, entities in the economic field are more generalized and constantly accompanied by the emergence of new entities; and second, the position of the entity is usually adjacent to the attribute and the POS is consistent with the attribute, which is common in noun combinations. Therefore, even if the entity is recognized, there is a high probability that it may be recognized as an attribute, thus resulting in poor results. Obviously, the entity recognition rate is slightly improved under our method as shown in Figure 4 because the attribute recognition rate is improved, which effectively reduces the interference with entity recognition.

In Table 5, the attribute (F = 64.19) and the attribute value (F = 74.62) were overall optimistic under the LSTM-CRF model. Because the attributes and attribute values were mostly composed of phrases and clauses while the LSTM model has the characteristics of long-distance dependence, the LSTM-CRF model is suitable for the recognition of long elements. However, when using the POS syntactic rules method, the precision of attributes and attribute values (P = 79.72, P =8 1.55) was slightly higher than that of the LSTM-CRF (P = 71.35, 76.81) because we directly used the syntactic dependencies between words. The main idea of the integrity algorithm is to use the rule recognition results to correct the recognition boundary of the LSTM-CRF model and further improve the performance, which can be clearly observed in Figures 5 and 6.

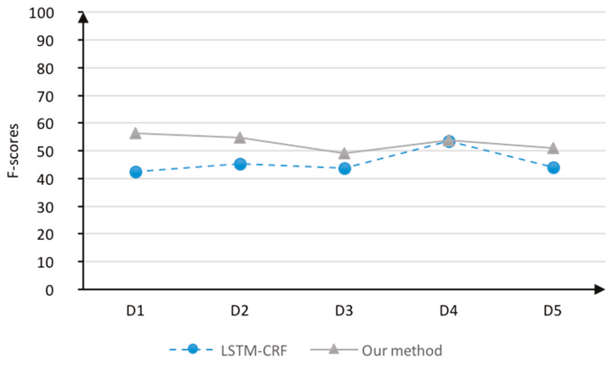


**Figure 4.** Comparison of F-scores of entity recognition in the five data sets.

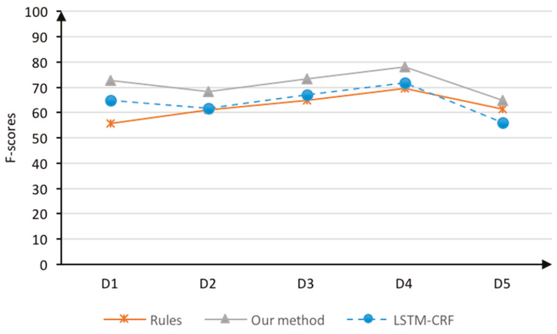


**Figure 5.** Comparison of the F-scores of attributes recognition in the five data sets.

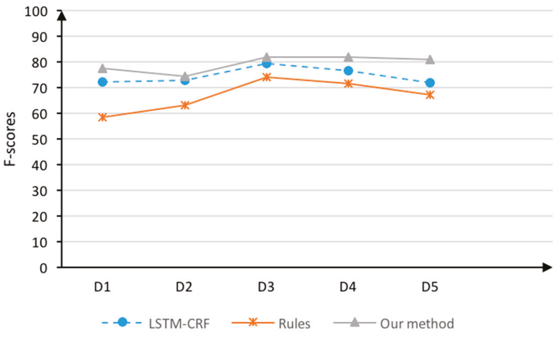


**Figure 6.** Comparison of the F-scores of attribute value recognition in the five data sets.

### 4.4.2. The Field Cross-Recognition Results

To further verify the effectiveness of the LSTM-CRF model with the integrity algorithm and determine whether the algorithm displays good domain independence, the corpus of the pharmaceutical industry is used as a training set. The corpus of the medical industry, which is similar to that for the pharmaceutical industry, is used as test set 1, and the corpus of the car manufacturing industry, which is not associated with the pharmaceutical industry, is selected as test set 2. The results are shown in Tables 6–8, respectively.

A comparison of the recognition results of test set 1 and test set 2 in Table 8 show that the F-scores of the test set 1 entities and attributes were higher than that of the test set 2 under the accurate evaluation, whereas the F-scores of the attribute values were close to each other, although test set 2

(F = 78.77) was slightly higher than test set 1 (F = 77.67). Entities and attributes were domain-related. An entity is a unique name in a domain, and an attribute is the characteristics of the entity; therefore, the test set 1 entity and attribute recognition performances were slightly better. The attribute value was not relevant to the domain and represents the scope of an attribute, and its constituent elements often have similarities. Therefore, the recognition of results for the attribute values of test set 1 and test set 2 were close to each other, and test set 2 may even be slightly higher than test set 1.

A comparison of the Avg (average value) of Table 8 with the Avg of each element in our method in Table 5 shows that the recognition results of our method were indeed higher than that shown in Table 5 because the LSTM-CRF model had field adaptability. Although the two sets of experiments prove that the cross-domain recognition performance was not as good as the training performance in the same field, the overall result shows that the recognition effect was not much different, which indicates that our method has a certain degree of versatility.

Finally, a comparison of the results in Tables 6–8 shows that the F-scores of the entities, attributes, and attribute values in our model are much higher than that of the LSTM-CRF model and POS syntactic rules. This finding again shows that our method recognizes phrase elements and clause-level elements and thus can effectively improve the integrity of the elements. In addition, the experiment proves that our method also has advantages in other fields, which fully demonstrates that the model is domain independent.

**Table 6.** Field cross-recognition results for the LSTM-CRF model (%).

| Test Set | Elements | Accurate Evaluation | | | Coverage Evaluation | | |
|---|---|---|---|---|---|---|---|
| | | P | R | F | P | R | F |
| 1 | En | 56.25 | 36.00 | 43.90 | 65.35 | 58.06 | 61.49 |
| | Attr | 70.15 | 46.53 | 55.95 | 77.34 | 54.87 | 64.20 |
| | Val | 71.83 | 71.83 | 71.83 | 80.82 | 76.39 | 78.20 |
| 2 | En | 56.92 | 43.79 | 49.50 | 63.32 | 51.80 | 56.94 |
| | Attr | 59.58 | 41.94 | 49.23 | 73.25 | 48.62 | 58.45 |
| | Val | 70.25 | 67.82 | 69.01 | 83.18 | 76.03 | 79.44 |
| Avg | En | 56.59 | 39.90 | 46.70 | 64.33 | 54.93 | 59.22 |
| | Attr | 64.87 | 44.24 | 52.59 | 75.30 | 51.75 | 61.33 |
| | Val | 71.04 | 69.83 | 70.42 | 82.00 | 76.21 | 78.82 |

**Table 7.** Field cross-recognition results for the rules (%).

| Test Set | Elements | Accurate Evaluation | | | Coverage Evaluation | | |
|---|---|---|---|---|---|---|---|
| | | P | R | F | P | R | F |
| 1 | Attr | 74.28 | 40.13 | 52.11 | 75.52 | 43.37 | 55.10 |
| | Val | 78.60 | 53.74 | 63.84 | 82.48 | 60.44 | 69.76 |
| 2 | Attr | 63.21 | 37.86 | 47.36 | 70.14 | 44.90 | 54.75 |
| | Val | 72.93 | 50.54 | 59.70 | 77.36 | 53.02 | 62.92 |
| Avg | Attr | 68.75 | 39.00 | 49.74 | 72.83 | 44.14 | 54.93 |
| | Val | 75.77 | 52.14 | 61.77 | 79.92 | 56.73 | 66.34 |

**Table 8.** Field cross-recognition results for the Our method (%).

| Test Set | Elements | Accurate Evaluation | | | Coverage Evaluation | | |
|---|---|---|---|---|---|---|---|
| | | P | R | F | P | R | F |
| 1 | En | 57.35 | 41.20 | 47.95 | 65.21 | 50.82 | 57.12 |
| | Attr | 80.62 | 58.23 | 67.62 | 85.14 | 67.44 | 75.26 |
| | Val | 83.10 | 72.91 | 77.67 | 88.62 | 75.23 | 81.38 |
| 2 | En | 53.37 | 39.50 | 45.40 | 61.67 | 45.20 | 52.17 |
| | Attr | 69.93 | 43.46 | 53.61 | 79.95 | 61.74 | 69.67 |
| | Val | 78.03 | 79.52. | 78.77 | 84.29 | 80.59 | 82.40 |
| Avg | En | 55.36 | 40.35 | 46.68 | 63.44 | 48.01 | 54.65 |
| | Attr | 75.28 | 50.85 | 60.62 | 82.55 | 64.59 | 72.47 |
| | Val | 80.57 | 76.25 | 78.22 | 86.46 | 77.91 | 81.89 |

### 4.4.3. Comparative Experiments

To further validate the effectiveness of our approach, we specifically compare our methods with several advanced methods for attribute. The test results are as seen in Table 9.

**Table 9.** Comparative experiments of attribute (%). The F value is calculated under the precise evaluation.

| Model | F |
|---|---|
| CRF | 60.65 |
| LSTM-CRF | 64.19 |
| Char-LSTM-CRF | 72.56 |
| LSTM-CNNs-CRF | 73.09 |
| Our method | 72.87 |
| LSTM-CNNs-CRF + Integrity algorithm | 75.21 |

CRF [23]: The CRF model was proposed by Lafferty. It incorporates the word, POS, n-gram words, suffix and prefix, and location features.

LSTM-CRF [16]: In article, the LSTM-CRF model is like that proposed by Huang et al. [16].

Char-LSTM-CRF [18]: The character embedding of the words is given to a bidirectional LSTM. Finally, it concatenates outputs to an embedding from a lookup table to obtain a representation for this word.

LSTM-CNNs-CRF [24]: The CNN training obtains the character embedding, then connects the character embedding and the word embedding, and inputs it into the LSTM. Finally, it inputs the vector output from the LSTM into the CRF to jointly decode to obtain the optimal sequence label.

As can be seen from Table 9, our method is significantly superior to the basic methods of CRF and LSTM-CRF. Compared to the Char-LSTM-CRF and LSTM-CNNs-CRF models, we consider syntactic features, which works well. In addition, we can also see that combining LSTM-CNNs-CRF with integrity algorithm can achieve good results. This is because the integrity algorithm can analyze semantics from a language perspective and improve the integrity of element extraction. This again illustrates the validity and migration of the integrity algorithm.

## 5. Conclusions

In summary, our research focuses on firm reports in the financial field. FLSs are used as objects to recognize valuable financial information, such as entities, attributes, and attribute values. Considering the three different types of elements, a synchronous recognition strategy with the advantages of dependency syntax is incorporated to capture the structure of elements and define POS syntactic rules based on the contexts of attributes and attribute values. Then, the integrity algorithm is used to correct the boundaries of the LSTM-CRF model labeling results. Finally, without losing the recall rate, the accuracy of the model is improved by correcting the integrity of the element, thereby optimizing the model performance. In addition, experiments in different fields are repeated. The experiments showed that the proposed model displays good domain independence and can be easily applied in various fields. Integrity algorithms can also be easily combined with neural network models to avoid relying solely on being data driven.

The next steps in this research are as follows. First, we will continue to conduct research on the boundaries of elements and further improve the effectiveness of recognition. Second, because information on the Internet is both genuine and fake, methods of distinguishing between genuine and fake information and selecting high-value information should be investigated. Third, determining how to use the identified elements to interpret the current status of a company and provide decision support and early warnings will be a focus of upcoming research.

**Author Contributions:** Methodology, experimental analysis, and paper writing, R.G.; The work was done under the supervision and guidance of D.X. and Z.N.

**Funding:** This work is sponsored by Natural Science Foundation of Shanghai, Project Number: 16ZR1411200.

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

## References

1. Ding, X.; Zhang, Y.; Liu, T.; Duan, J. Deep learning for event-driven stock prediction. In Proceedings of the International Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015; pp. 2327–2333.
2. Lee, H.; Surdeanu, M.; MacCartney, B.; Jurafsky, D. On the importance of text analysis for stock price prediction. In Proceedings of the LREC 2014, Reykjavik, Iceland, 26–31 May 2014; pp. 1170–1175.
3. Qiu, X.Y.; Srinivasan, P.; Hu, Y. Supervised learning models to predict firm performance with annual reports: An empirical study. *J. Assoc. Inf. Sci. Technol.* **2014**, *65*, 400–413. [CrossRef]
4. Kumar, A.; Sethi, A.; Akhtar, M.S.; Ekbal, A.; Biemann, C.; Bhattacharyya, P. IITPB at SemEval-2017 Task 5: Sentiment Prediction in Financial Text. In Proceedings of the International Workshop on Semantic Evaluation, Vancouver, BC, Canada, 3–4 August 2017; pp. 894–898.
5. Kumar, A.; Alam, H.; Werner, T.; Vyas, M. Experiments in Candidate Phrase Selection for Financial Named Entity Extraction-A Demo. In Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations, Osaka, Japan, 11–16 December 2016; pp. 45–48.
6. Jiang, T.Q.; Wan, C.X.; Liu, D.X. Evaluation Object-Emotional Word Pair Extraction Based on Semantic Analysis. *Chin. J. Comput.* **2017**, *40*, 617–633.
7. Li, F. The information content of forward-looking statements in corporate filings—A naïve Bayesian machine learning approach. *J. Account. Res.* **2010**, *48*, 1049–1102. [CrossRef]
8. Feldman, R.; Fresco, M.; Goldenberg, J.; Netzer, O.; Ungar, L. Extracting product comparisons from discussion boards. In Proceedings of the IEEE International Conference on Data Mining, Omaha, NE, USA, 28–31 October 2007; pp. 469–474.
9. Hu, M.; Liu, B. Mining and summarizing customer reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; pp. 168–177.
10. Qiu, G.; Liu, B.; Bu, J.; Chen, C. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.* **2011**, *37*, 9–27. [CrossRef]
11. Chieu, H.L.; Ng, H.T. A maximum entropy approach to information extraction from semi-structured and free text. In Proceedings of the Eighteenth National Conference on Artificial Intelligence, Edmonton, AB, Canada, 28 July–1 August 2002; pp. 786–791.
12. Zhang, S.; Jia, W.J.; Xia, Y.; Meng, Y.; Yu, H. Extracting Product Features and Sentiments from Chinese Customer Reviews. In Proceedings of the 7th LREC, Valletta, Malta, 17–23 May 2010; pp. 17–23.
13. Finkel, J.R.; Grenager, T.; Manning, C. Incorporating non-local information into information extraction systems by Gibbs sampling. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Ann Arbor, MI, USA, 25–30 June 2005; pp. 363–370.

14. Choi, Y.; Cardie, C.; Riloff, E.; Patwardhan, S. Identifying sources of opinions with conditional random fields and extraction patterns. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, Vancouver, BC, Canada, 6–8 October 2005; pp. 355–362.
15. Pinheiro, P.H.; Collobert, R. Recurrent convolutional neural networks for scene parsing. *arXiv*, **2013**, arXiv:1306.2795.
16. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv*, **2015**, arXiv:1508.01991.
17. Limsopatham, N.; Collier, N.H. Bidirectional LSTM for named entity recognition in Twitter messages. In Proceedings of the 26th International Conference on Computational Linguistics, Osaka, Japan, 11–16 December 2016; pp. 145–152.
18. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural architectures for named entity recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 260–270.
19. Popescu, A.M.; Etzioni, O. Extracting product features and opinions from reviews. In *Natural Language Processing and Text Mining*; Springer: London, UK, 2007; pp. 9–28.
20. Bloom, K.; Garg, N.; Argamon, S. Extracting appraisal expressions. In Proceedings of the Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Rochester, NY, USA, 22–27 April 2007; pp. 308–315.
21. Somprasertsri, G.; Lalitrojwong, P. Mining Feature-Opinion in Online Customer Reviews for Opinion Summarization. *J. Univ. Comput. Sci.* **2010**, *16*, 938–955.
22. Graves, A. *Supervised Sequence Labelling with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; ISBN 9783642212703.
23. Lafferty, J.; McCallum, A.; Pereira, F.C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning Morgan Kaufmann Publishers, San Francisco, CA, USA, 28 June–1 July 2001; pp. 282–289.
24. Ma, X.; Hovy, E. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *arXiv*, **2016**, arXiv:1603.01354.

*future internet*

MDPI

*Article*

# Object Detection Network Based on Feature Fusion and Attention Mechanism

**Ying Zhang [1], Yimin Chen [1,2,*], Chen Huang [1] and Mingke Gao [3]**

1 School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; youmemeda@shu.edu.cn (Y.Z.); channinghuang@shu.edu.cn (C.H.)
2 Shanghai Institute for Advanced Communication and Data Science, Shanghai 200444, China
3 The 32nd Research Institute, China Electronics Technology Group Corporation, No. 63 Chengliugong Road, Jiading District, Shanghai 200444, China; gaomingke@shu.edu.cn
* Correspondence: ymchen@mail.shu.edu.cn

Received: 9 November 2018; Accepted: 25 December 2018; Published: 2 January 2019

**Abstract:** In recent years, almost all of the current top-performing object detection networks use CNN (convolutional neural networks) features. State-of-the-art object detection networks depend on CNN features. In this work, we add feature fusion in the object detection network to obtain a better CNN feature, which incorporates well deep, but semantic, and shallow, but high-resolution, CNN features, thus improving the performance of a small object. Also, the attention mechanism was applied to our object detection network, AF R-CNN (attention mechanism and convolution feature fusion based object detection), to enhance the impact of significant features and weaken background interference. Our AF R-CNN is a single end to end network. We choose the pre-trained network, VGG-16, to extract CNN features. Our detection network is trained on the dataset, PASCAL VOC 2007 and 2012. Empirical evaluation of the PASCAL VOC 2007 dataset demonstrates the effectiveness and improvement of our approach. Our AF R-CNN achieves an object detection accuracy of 75.9% on PASCAL VOC 2007, six points higher than Faster R-CNN.

**Keywords:** CNN; object detection network; attention mechanism; feature fusion

## 1. Introduction

Recently, object detection accuracy has been improved by using deep CNN (Convolutional Neural Network) nets [1]. There are two types of object detection methods: Region-based object detection and regression-based object detection. Representative algorithms of regression-based object detection are R-CNN (region with CNN features) [2], SPP-net [3], Fast R-CNN [4], and Faster R-CNN [5]. Faster R-CNN is an end-to-end object detection network because it combines region proposal and detection into a unified network. The object detection networks have three essential components called the feature extraction net, region proposal network, and classification and regression.

Object detection always uses deep CNN nets (VGG-16(very deep convolutional networks)) [1], ResNet [6]), to extract CNN features due to the discriminative representations of CNN features. Deep CNN features contain high-level semantic information, which can better detect objects. However, deep CNN features lose a lot of detailed texture information because of high abstraction. Object detection is a more challenging task when objects are small. Object detection is difficult to achieve the desired effect due to illumination variations, occlusions, and a complex background. As mentioned above, features are crucial for object detection. Object detection often uses the last CNN feature maps for region proposal net. In [7], the author combined high layer information with low layer information for semantic segmentation. HyperNet [8], which combines the different CNN features, achieved the desired effect accuracy of object detection. The proper fusion of CNN features is more suitable for proposal generation and detection.

Recently, many studies have hoped to add attention mechanisms to deep networks. The attention model in deep learning actually simulates the attention model of the human brain. In 2015, Kelvin Xu et al. [9] introduced the attention in image caption. The visual attention mechanism can quickly locate the region of interest in the complex scene, finding the target of interest, and selectively ignoring the region of no interest. The visual attention mechanism is divided into two ways: Bottom-up model, which is also called hard attention, and top-down model, which is called soft attention. Anderson P et al. [10] applied hard attention to image caption. However, more research and applications are still more inclined to use soft attention because it can be directly derived for gradient back propagation. The paper [11] proposed the soft attention model and applied it to machine translation. The traditional attention mechanism is soft attention [11,12], which is obtained by deterministic score calculation to obtain the coded hidden state after the attainment. Soft attention is parameterized and it can be guided and embedded in the model for direct training. The gradient can be passed back through the attention mechanism module to other parts of the model. Object detection is difficult due to the complex background. Pixels are treated the same in the CNN feature maps and the region proposal net. The object is easily disturbed by the background, leading to inaccurate detection. Object detection networks need to enhance the impact of areas of interest and weaken interference from an unrelated background. So, we will apply the attention to the target detection depth network.

In this paper, we use the faster R-CNN object detection network as a framework. Deep network VGG-16 is still used in the feature extraction part. We propose a new object detection network that fixes the disadvantages of feature fusion and the attention mechanism in the faster R-CNN in case of background interference and small target problems. The improved object detection network (we called this AF R-CNN) is also an end-to-end network. Our main contributions are two-fold:

1. Feature fusion: We fuse the 3, 4, 5 layers of CNN feature maps by maximum pooling conv-3 and deconvoluting conv-5, which was extracted from VGG-16. The new CNN feature map combines fine, shallow layer information with coarse, deep layer information.
2. Attention mechanism: We propose a new network branch to generate a weight mask, which enhances the interest features and weakens the irrelevant feature in the CNN feature map. The attention network branch assigns attention weights to the CNN feature, making region proposal net more efficient and meaningful.

On the detection challenges of PASCAL VOC 2007, we achieved state-of-the-art mAP of 75.9% and 79.7%, outperforming the seminal Faster R-CNN by 6 and 6.5 points, correspondingly.

## 2. Related Works

### *2.1. Object Detection*

We review object detection methods in this section, especially deep learning based methods. Object detection is designed to localize and identify every object using a bounding box. In the detection framework, object proposals [13] reduce the computational complexity compared with sliding window methods [14]. The traditional methods use manual features, such as edges and shapes. Deep learning based methods use CNN as features. With the great success of the deep learning [1,6], two major object detection methods based on CNN have been proposed: Proposal based methods [5,15] and proposal free methods [16,17]. The Faster R-CNN and its variants [18,19] have been the dominating methods for years. Mask R-CNN [20] uses RolAlign (a simple, quantization-free layer) instead of Rol (region of interest) pooling in the Faster R-CNN. Wang et al. [19] proposed a confrontation network (ASTN, ASDN) for occlusion and deformation. For the small target problem of target detection, Li et al. [21] proposed PGAN (perceptual generative adversarial network) in the object detection framework. Faster R-CNN and its variants are proposal based methods. Different from these methods, YOLO (You Only Look Once) [17] and its variants predict bounding boxes and class probabilities directly from full images. YOLO 9000 [16], which achieves higher accuracy and speed, proposes a joint training strategy. YOLO and its variants are proposal free methods.

## *2.2. Visual Attention Mechanism*

Recently, neuroscience and cognitive learning theory have developed rapidly. Evidence from the human perception process shows the importance of the attention mechanism. Koch [22] proposed a neurobiology framework that laid the foundation for the visual attention model. The visual attention mechanism is divided into two processing methods: Bottom-up model and top-down model. The bottom-up visual attention model is driven by data from the low-level features of the image. However, the top-down visual attention model is more complicated than the bottom-up visual attention model. The top-down visual attention model is driven by tasks, so it requires tasks to provide relevant prior knowledge.

Various visual attention models have been proposed. Itti et al. [23] proposed the Iitti attention model, which is based on feature integration. Bruce et al. [24] put forward the ATM attention model by training with image sub-blocks. Ali Borji et al. [25] combined bottom-up and top-down models. Recently, tentative efforts have been made towards applying attention to the deep neural network. The attention model has good results in the fields of image caption, machine translation, speech recognition, etc. Bahdanau et al. [11] proposed a single-layer attention model to solve source language alignment problems of different lengths in machine translation. Wang [26] applied a single attention model to news recommendation and screening filed. Muti-attention mechanisms (hierarchical attention, dual attention) can accomplish tasks more accurately. Rijke [27] proposed the hierarchical attention model to complete the abstract extraction of the article. Seo et al. [28] used the dual attention model for the recommendation system.

In object detection, there are several methods based on attention mechanisms that have been proposed. NonlocalNet [29] used self-attention to learn the correlation between features on the feature map and obtain new features. Hu [12] proposed an object relation module based on self-attention. The visual attention mechanism can be roughly divided into two types: Learning weight distribution and task focus. We added the attention module in the classification network to the learn weight. The attention module is a mask branch to enhance the interest feature and weak background interference.

## 3. Methods

Our object detection system, AF R-CNN, was composed of four modules: A deep convolutional network, VGG-16, that extracts features, feature fusion, and the attention modules, and a Faster R-CNN detector that uses the region proposal network (RPN). The whole object detection system is a unified network. Our AF R-CNN is illustrated in Figure 1. First, AF R-CNN produces feature maps through the VGG-16 net. We fused feature maps, which came from different layers, and then compressed them into a uniform feature map. Next, we obtained the weight from the attention module, and the RPN produced proposals. Finally, these proposals were classified and regressed.

### *3.1. Deep Convolutional Network:VGG-16 Net*

Karen Simonyan proposed a series of VGG-16 networks in the paper [1]. The VGG-16 network contains 13 convolution layers and three full connect layers, as shown in Figure 2. In the convolution layers, there are 13 convolution layers, 13 relu (Rectified Liner Units) layers, and four pooling layers. Figure 2 shows the architecture of the VGG-16 network.

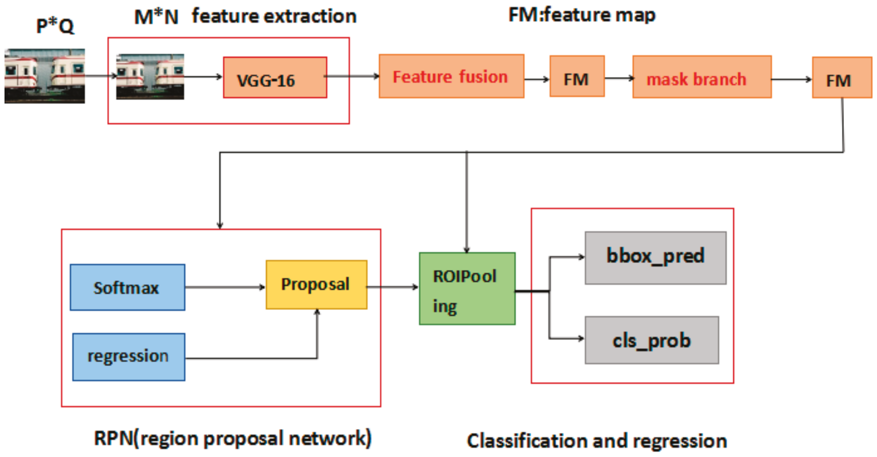
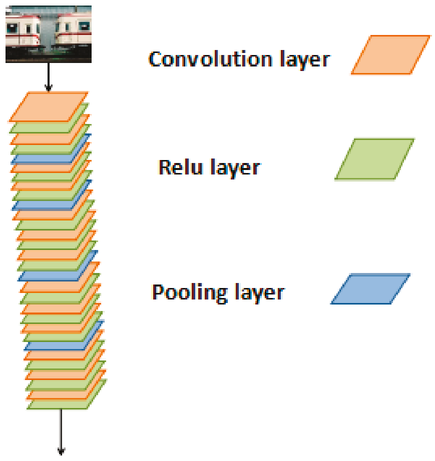
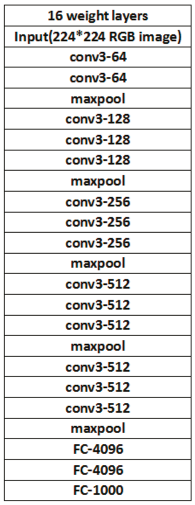


**Figure 1.** The architecture of AF R-CNN. The red part is the module we added above the original structure. The AF R-CNN is composed of four modules: A deep convolutional network, VGG-16, that extracts features, feature fusion, and the attention modules, and a Faster R-CNN detector that uses the region proposal network (RPN).



(**a**) Structure diagram



| 16 weight layers |
|---|
| Input(224*224 RGB image) |
| conv3-64 |
| conv3-64 |
| maxpool |
| conv3-128 |
| conv3-128 |
| conv3-128 |
| maxpool |
| conv3-256 |
| conv3-256 |
| conv3-256 |
| maxpool |
| conv3-512 |
| conv3-512 |
| conv3-512 |
| maxpool |
| conv3-512 |
| conv3-512 |
| conv3-512 |
| maxpool |
| FC-4096 |
| FC-4096 |
| FC-1000 |

(**b**) Detailed data information

**Figure 2.** The architecture of VGG-16. (**a**) shows the structure diagram of the VGG-16 net; (**b**) shows the detailed data information of the VGG-16 net. In the convolution layers, there are 13 convolution layers, 13 relu (Rectified Liner Units) layers, and four pooling layers.

In all convolution layers, the convolution kernel is 3 × 3 and stride is 1. In the Faster R-CNN, the size of the image becomes (m + 2)(n + 2) because all convolution features are padded (pad = 1). The advantage of this is that the output image's size is the same as the input. The convolution formula is given by:

$$output_{size} = \frac{(input_{size}) - kernel_{size} + 2 \times pad}{stride} + 1 \tag{1}$$

where $input_{size}$ represents the size of the image or the size of the feature map in the middle. The $output_{size}$ represents the size of the feature map after the convolution kernel. The $kernel_{size}$ donates the size of the convolution kernel.

Each pooling layer will make the size of the output one-fourth of the input. After four pooling layers, the size of the convolution feature is $(m/16)(n/16)$.

In Figure 3, the detailed padding process is shown. In the picture on the left, the blue area represents the size of the input image ($(M \times N)$). The red area represents the size of the convolution kernel ($3 \times 3$). After padding(pad=1), the image size becomes $(M + 2) \times (N + 2)$. Additionally, we can see in the picture on the right, the size of the convolution features are the same as the input picture after the convolution kernel. This is the meaning of the padding.

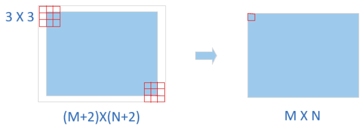


**Figure 3.** The process of padding. The size of the input image is $(M \times N)$. After padding(pad=1), the image size becomes $(M + 2) \times (N + 2)$.

## 3.2. Faster R-CNN Detector

Faster R-CNN combines the feature extraction net, region proposal net, and classification and regression into a network. Recently, many methods have been proposed based on Faster R-CNN. Figure 4 illustrates the Faster R-CNN architecture.

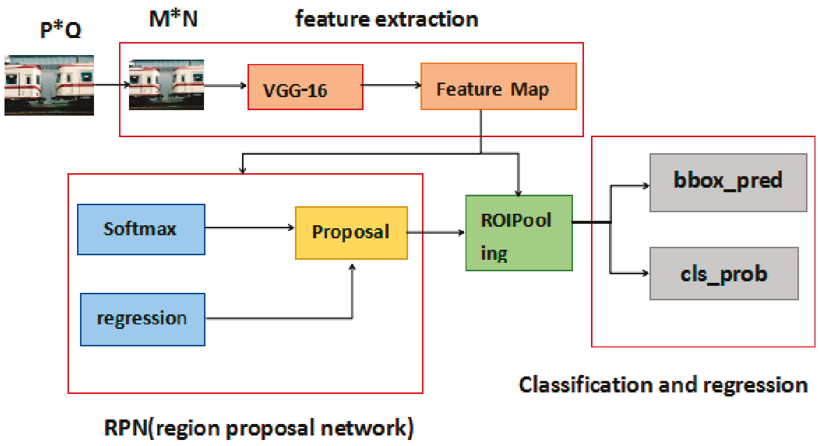


**Figure 4.** The architecture of Faster R-CNN. Faster R-CNN combines feature extraction net, region proposal net, and classification and regression into a network.

From Figure 4 we can see that Faster R-CNN architecture contains four main modules:

***Feature extraction network.*** Faster R-CNN uses the VGG-16 net to generate convolution feature maps. We know that Faster R-CNN supports any size of input images because the images are normalized before entering the network. We assume that the normalized image size is m*n. After the convolution layers in the VGG-16 net, the size of the image changes to $(m/16) \times (n/16)$. The convolution feature map is $(m/16) \times (n/16) \times 512$, which is shared for the region proposal network (RPN) and fully connected layers.

***Region Proposal Network (RPN).*** R-CNN and Faster R-CNN use selective search (SS) as external modules independent of the object detection architecture. Object proposal methods include methods based on grouping super-pixels [30] and methods based on sliding windows [31]. Faster R-CNN uses the region proposal network (RPN) to generate detection proposals. The anchor mechanism is the most important part of RPN. An anchor is centered at the sliding window in question. Multiple scale windows are required due to different object sizes and length to width ratios. The anchor gives a reference window size that different sizes of windows are obtained in three scales and three aspect ratios. In the region proposal network (RPN), the feature map convolutes with a $3 \times 3$ sliding window first. Then, the output divides into two ways by the $1 \times 1$ convolution kernel. The softmax function is used to classify anchors (foreground or background). The offset of anchors' bounding box regression can also be obtained to get the precise proposal. The two network branches are combined to obtain the proposal regions.

***ROI (Region Of Interest) Pooling.*** The RoI pooling layer obtains a fixed size of the proposal feature map using the proposal regions and feature map in the RPN and VGG-16.

***Classification and regression.*** The proposal feature map calculates the probability vector, which represents the category of proposals by the full connect layer and softmax function. Additionally, it obtains the position offset to get a more accurate proposal by using bounding box regression.

The steps of Faster R-CNN are as follows:

1. Input test image;
2. Extract about a 2000 proposal region from the image using the selective search algorithm;
3. Input the image into the VGG-16 net for feature extraction;
4. Mapp the proposal region to the feature map;
5. Generate a fixed size feature map in the RoI pooling layer; and
6. Perform classification and bounding box regression.

### *3.3. Feature Fusion*

Faster R-CNN extracts features through a convolutional neural network to generate feature maps for RPN and object classification and border regression. The quality of the feature map greatly affects the performance of the entire object detection algorithm. The Faster R-CNN uses the VGG-16 network as the basic network and produces a final feature map after the fifth convolution layer. Although the feature map obtains high-level semantic information through deep convolution layers, the high-level feature map loses a lot of detailed texture information due to high abstraction, resulting in inaccurate positioning of objects and difficulties in the detection of small target objects.

To solve the above problems, the AF R-CNN model we proposed in this paper fuses features from different convolutional layers. Given a picture, AF R-CNN uses the pre-trained model, VGG-16, to compute feature maps. Shallow and Deep CNN features are complementary for object detection. Figure 5 shows the detailed process of feature fusion. We fused feature maps from the third, fourth, and fifth convolution layers (con_3, con_4, con_5). These feature maps have different resolutions because of subsampling and pooling operations. To get the feature maps of the same resolution, we adopted different sampling strategies. For the shallow layer, we added the maximum pooling to conduct subsampling. Additionally, we added the deconvolutional operation to carry out upsampling in the deep layer. The fourth convolutional layer remained at the original size. Then, we normalized multiple feature maps using local response normalization (LRN) [32]. For each feature map, an additional convolution layer was required. The feature maps were connected by the connection operation to form a new feature map. The new feature map contained shallow and deep layer information that was effective for object detection. The feature map resolution was more suitable for detection.

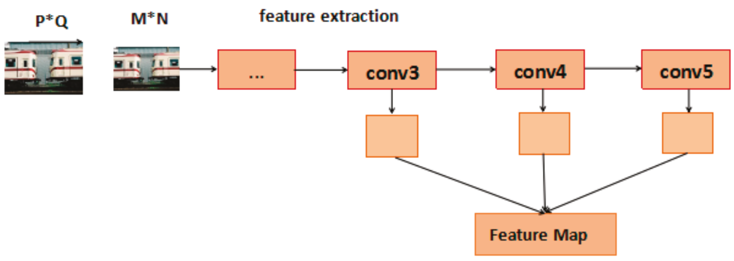


**Figure 5.** Detailed process of feature fusion. We fused feature maps from the third, fourth, and fifth convolution layers(con_3, con_4, con_5).

## *3.4. Visual Attention Mechanism*

AF R-CNN uses the two parallel branches of the trunk and the attention branch to generate attention-aware features and then select regions on the feature map after adding an attention mask. Our attention modules were constructed by a mask branch. Given the feature map, $T(x)$, as the input, the mask branch learns the mask, $M(x)$ by using the bottom-up top-down structure [7,33–35]. The mask, $M(x)$, is used as control gates for $T(x)$. The attention module $H(x)$ is defined as:

$$H_{i,c}(x) = M_{i,c}(x) \times T_{i,c}(x) \tag{2}$$

where $i$ ranges over all spatial positions and $c$ represents channels.

Figure 6 display the architecture of the mask branch. The mask branch contains two steps: Feed-forward sweep and top-down feedback. The two steps behave as a bottom-up top-down fully convolutional structure. The feed-forward sweep operation extracts global information quickly. Additionally, the top-down feedback operation feeds back the global information to the feature map. Firstly, max pooling was performed in the fusion feature map to improve the receptive field. The global information, which was expanded by the top-down structure, guides features when reaching the lowest resolution. The extended structure up samples global prior information by bilinear interpolation. To make the size of the output feature map the same as the input feature map, the output dimension of the bilinear interpolation was the same as the max pooling. Then, the attention module used the sigmoid layer to normalize the output results, with $M(x)$. $M(x)$ ranging from 0 to 1. This was the attention weight of the feature map at each location. This weight matrix, $M(x)$, and the feature map matrix, $T(x)$, were multiplied to obtain the desired attention-weighted feature map.

The structure of this encoder-decoder (bottom-up top-down) is often used in image segmentation, such as FCN [7]. We applied a bottom-up top-down structure to the object detection net. The structure of bottom-up top-down first extracted high-level features and increased the receptive field of the model through a series of convolution and pooling. Pixels activated in the high-level feature map can reflect the area where attention is located. The size of the feature map was enlarged by the up sample to the same size as the original input, so that the area of the attention was mapped to each pixel of the input. We called this the attention map, ($M(x)$). Each pixel value in the attention map ($M(x)$) output by the soft mask branch was equivalent to the weight of each pixel value on the original feature map, which enhanced the meaningful features and suppressed meaningless information. This weight matrix, $M(x)$, and the feature map matrix, $T(x)$, were multiplied to obtain the desired attention-weighted feature map, ($H(x)$).

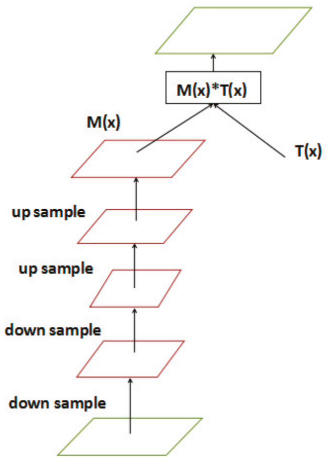


**Figure 6.** The architecture of the mask branch. The mask branch contains two steps: Feed-forward sweep and top-down feedback. The two steps behave as a bottom-up top-down fully convolutional structure.

The object detection module shares the convolution parameters with the attention module to achieve an integrated end-to-end object detection network. The mask branch aims at improving features rather than solving complex problems directly. It worked as a feature selector, which enhances good features and suppress noises from features.

### *3.5. AF R-CNN Loss Function*

We minimize an objective function following the loss in the Faster R-CNN. The loss function in AF R-CNN is defined as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) \tag{3}$$

Hence, $i$ represents the index of the anchor. The $p_i$ is the predicted probability whether anchor, $i$, is an object and $t_i$ is a vector of four parameters in the predicted bounding box. The classification loss, $L_{cls}$, is the loss over two classes (object vs. not object). We used $N_{cls}$ and $N_{reg}$ to normalize two terms, cls(classification) and reg(regression).

The ground-truth label is defined as follows:

$$p_i^* = \begin{cases} 1, \; the\ anchor\ is\ positive \\ 0, \; the\ anchor\ is\ negative \end{cases} \tag{4}$$

The classification and regression loss are defined as:

$$L_{reg}(t_i,t_i^*) = R(t_i - t_i^*) \tag{5}$$

where, $R$ is the robust loss, which is defined as follows:

$$R(x) = \begin{cases} 0.5x^2 if\ |x| < 1 \\ |x| - 0.5 otherwise \end{cases} \tag{6}$$

We set $\lambda = 10$, so that the weight of the classification and regression terms were roughly equal. The $t_i$ is defined as follows:

$$t_x = \frac{(x - x_a)}{w_a}, t_y = \frac{(y - y_a)}{h_a} \tag{7}$$

$$t_w = log(\frac{w}{w_a}), t_h = log(\frac{h}{h_a}) \tag{8}$$

$$t_x^* = \frac{(x^* - x_a)}{w_a}, t_y^* = \frac{(y^* - y_a)}{h_a} \tag{9}$$

$$t_w^* = log\left(\frac{w^*}{w_a}\right), t_h^* = log\left(\frac{h^*}{h_a}\right) \tag{10}$$

where $x$, $y$, $w$, and $h$ are the center coordinates of the box and width and height. $x_a(y_a, w_a, h_a)$ represents the same mean in the anchor box and $x^*(y^*, w^*, h^*)$ represents the ground-truth box.

## 4. Experiments

We performed a series of comprehensive experiments on the dataset, PASCAL VOC 2007 [36] and 2012. The PASCAL VOC 2007 dataset contains 20 object categories. The pixel size of the image varies, and is usually (horizontal view) 500 × 375 or (longitudinal view) 375 × 500. The mean average precision was used to measure the performance of the object detection network. All experiments were performed on a ThinkStation P510 PC with Intel(R) Xeon(R) E5-2623 2.6GHZ CPU and Quadro M5000 GPU with an 8 GB memory. Our experiments were implemented in Python with TensorFlow.

In our experiments, the AF R-CNN was trained on a training set and the detection results were obtained on the test set (PASCAL VOC 2007 test set). Training data 07 represents VOC 2007 trainval and 07+12 represents the union set of VOC 2007 trainval and VOC 2012 trainval. AF R-CNN and faster R-CNN use the same pre-trained model, VGG-16, which has 13 convolutional layers and three fully-connected layers. All the networks were trained using stochastic gradient descent (SGD) [37]. The initial learning rate was set to 0.001. As AF R-CNN, we re-scaled the images and resized the short side to 600. The initial image needed to be processed. The size of the image was fixed to 224 × 224 by mapping the image. Then, the fixed size image was input into the VGG network for feature extraction. We compared the AF R-CNN and faster R-CNN for object detection on the dataset, PASCAL VOC 2007.

Our experiments contained two parts. In the first part, we analyzed the effectiveness of each component in the AF R-CNN, including feature fusion and the soft mask branch in the attention module. After that, we compared our network with state-of-the-art results in the PASCAL VOC 2007 dataset.

Table 1 shows the results of the different object detection network. The first and second columns represent the results of the Faster R-CNN on the dataset, PASCAL VOC 2007. The third(fourth) column shows the results of the method, which we only added the attention module (feature fusion) in the Faster R-CNN. Additionally, the last two columns show the results of AF R-CNN, which we proposed in the article that contained both feature fusion and the attention module.

To understand the benefit of the attention mechanism, we calculated the mean average precision of the object. The attention module in the AF R-CNN is designed to suppress noise while keeping useful information by applying the dot product between the feature map and soft mask. From Table 1, we can see that the performance of the object detection network with the addition of the attention module was improved. In the experiments, we evaluated the effectiveness of the attention mechanism. The Faster RCNN+attention achieved an mAP (mean average precision) of 70.3%, 0.4 points higher than the Faster R-CNN. To understand the benefit of the attention mechanism, we calculated the mAP (mean average precision) of 20 object categories. The attention mechanism can enhance the feature contrast. The attention-aware features were more effective for significant objects, such as the car, boat, and person. However, for small targets, the detection efficiency was slightly reduced, such as the bottle, bird, and plant. This is because the attention mechanism was more effective for significant objects. However, for less significant targets, such as smaller targets and shallower targets, the attention mechanism did not achieve the desired results.

**Table 1.** Results of the Faster R-CNN and AF R-CNN on the dataset, PASCAL VOC 2007.

| | **Faster R-CNN** | **Faster R-CNN** | **Faster R-CNN+Attention** | **Faster R-CNN+Fusion** | **Ours** | **Ours** |
|---|---|---|---|---|---|---|
| Training data | 07 | 07+12 | 07 | 07 | 07 | 07+12 |
| mAP | 69.9 | 73.2 | 70.3 | 75.0 | 75.9 | 79.7 |
| areo | 70.0 | 76.5 | 71.4 | 73.6 | 76.2 | 83.0 |
| bike | 80.6 | 79.0 | 81.4 | 82.4 | 81.2 | 87.0 |
| bird | 70.1 | 70.9 | 63.9 | 75.1 | 77.9 | 81.4 |
| boat | 57.3 | 65.5 | 60.5 | 62.3 | 66.2 | 74.0 |
| bottle | 49.9 | 52.1 | 47.8 | 60.2 | 62.8 | 68.5 |
| bus | 78.2 | 83.1 | 79.5 | 80.2 | 80.2 | 87.7 |
| car | 80.4 | 84.7 | 81.5 | 83.3 | 86.3 | 88.0 |
| cat | 82.0 | 86.4 | 82.1 | 83.6 | 87.5 | 88.1 |
| chair | 52.2 | 52.0 | 50.3 | 59.3 | 56.9 | 62.4 |
| cow | 75.3 | 81.9 | 75.8 | 77.2 | 85.1 | 86.8 |
| table | 67.2 | 65.7 | 67.6 | 74.5 | 71.3 | 70.8 |
| dog | 80.3 | 84.8 | 81.6 | 84.8 | 87.2 | 88.6 |
| horse | 79.8 | 84.6 | 81.8 | 86.5 | 86.2 | 87.3 |
| mbike | 75.0 | 77.5 | 76.1 | 78.4 | 80.3 | 83.8 |
| person | 76.3 | 76.7 | 77.8 | 80.9 | 79.6 | 82.8 |
| plant | 39.1 | 38.8 | 35 | 53.8 | 47.3 | 53.2 |
| sheep | 68.3 | 73.6 | 68.8 | 70.4 | 77.3 | 81.1 |
| sofa | 67.3 | 73.9 | 67.8 | 72.2 | 75.2 | 77.6 |
| train | 81.1 | 83.0 | 83.3 | 83.2 | 79.1 | 84.3 |
| tv | 67.6 | 72.6 | 69.8 | 74.6 | 74.8 | 79.2 |

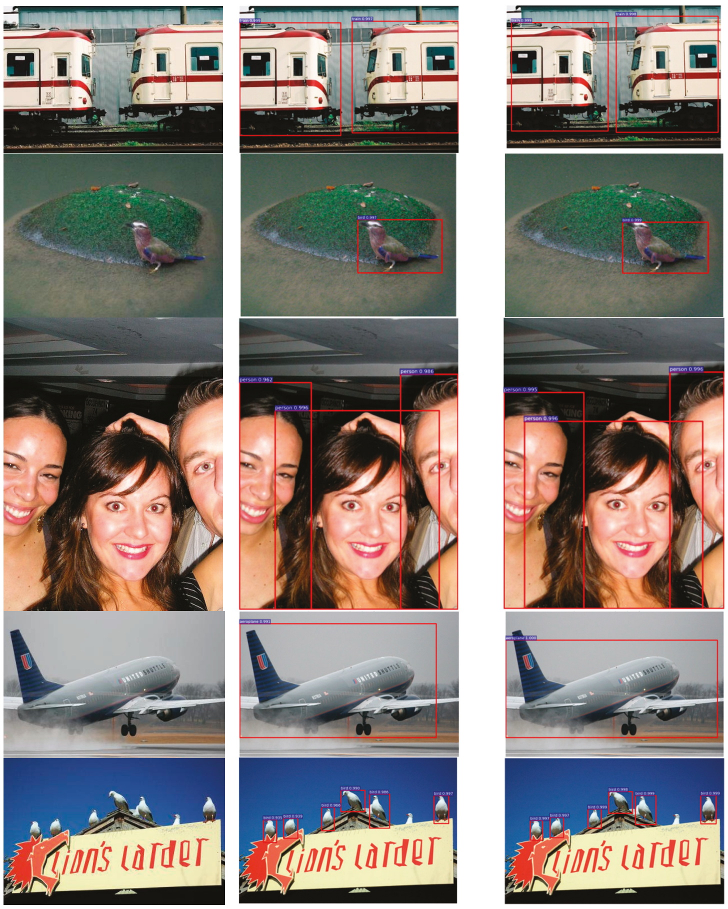
The mAP of the detection network Faster R-CNN+fusion was 75.0%, which was 5.1% higher than the Faster R-CNN on the dataset, PASCAL VOC 2007. As we have shown above, this is because the fusion features were more accurate than the deep convolution features. The results benefitted from more informative features. The reasonable resolution of features made for better object detection, especially when the object size was small. Our detection network outperformed the Faster R-CNN when the object size was small. For the plant, Faster R-CNN+fusion achieved a 53.8% mAP, a 14.7 points improvement, and for the bottle, the Faster R-CNN+fusion achieved a 60.2% mAP, 10.3 points higher than the Faster R-CNN. It showed that the multi-layer feature fusion could effectively improve the detection of small targets.

Our AF R-CNN outperformed the Faster R-CNN regardless of the object size. We compared our AF R-CNN with state-of-the-art methods, including Faster R-CNN [5], A-Fast-RCNN [19], CRAFT [38], and SSD [39] on the dataset, PASCAL VOC 2007. The results are shown in Table 2. Our AF R-CNN outperformed all methods on the PASCAL VOC 2007 dataset. The object detection network that we proposed in this article achieved an mAP of 75.9% on the dataset, PASCAL VOC 2007, 6 points higher than the Faster R-CNN because of feature fusion and the attention module. The above results suggest that our method enjoyed high efficiency and good performance.

**Table 2.** A comparison between the proposed method with the existing models on the dataset, PASCAL VOC 2007.

| **Method** | **Faster R-CNN** | **A-Fast-RCNN [28]** | **CRAFT [37]** | **SSD [38]** | **Ours** |
|---|---|---|---|---|---|
| VOC 2007 | 69.9 | 71.4 | 75.7 | 71.6 | 75.9 |

Figure 7 shows the visualization of the object detection. Left: The input images. Middle: The object detection of the Faster R-CNN. Right: The object detection of AF R-CNN. We found a few more representative pictures. First, in order to see the effect of our method on the object size, we selected a big plant (the fourth row) and a small bird (the second row). To assess the effectiveness of our approach to multiple goals and target defects, we chose the pictures in the first, third, and fifth cows. Of course, these pictures had crossovers, such as the last picture had many objects and the objects were very small. As can be seen from the results shown in Figure 7, the effect of the overall network was improved. The attention module helped detect significant targets while the feature fusion could effectively detect small targets. The experiments proved that our proposed method was more effective than Faster R-CNN.



(**a**) Original images (**b**) Faster R-CNN results (**c**) AF R-CNN results

**Figure 7.** Results of object detection. (**a**) shows the input images of the object detection networks; (**b**) shows the visualization of the detection results of the Faster R-CNN; (**c**) shows the visualization of the detection results of the AF R-CNN.

## 5. Conclusions

In this paper, we proposed AF R-CNN, a fully trainable deep architecture for object detection. AF R-CNN provides an efficient combination of the attention module and feature fusion for a deep object network. Our methods enhanced the impact of salient features and combined deep, but semantic, and shallow, but high-resolution, CNN features effectively. Thus, AF R-CNN improved the overall object detection accuracy.

However, our model still needs to be improved in terms of speed and real-time. How to balance the computational complexity and performance remains a big challenge. In the future, we would like to discover a lower computational burden and system complexity. Also, better pre-trained models, like res-net, will be applied to the research with the development of deep networks.

**Author Contributions:** Methodology, Experimental analysis and Paper Writing, Y.Z.; Writing-review and Data analysis, Y.Z. and Y.C.; Data and Writing Correction, C.H. and M.G.; The work was done under the supervision and guidance of Y.C.

**Funding:** This work is partially supported by Shanghai Innovation Action Plan Project (No. 16511101200) of Science and Technology Committee of Shanghai Municipality.

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

## References

1. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
2. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
3. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In Proceedings of the European Conference on Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014.
4. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
5. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
6. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *arXiv*, 2015; arXiv:1512.03385.
7. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
8. Kong, T.; Yao, A.; Chen, Y.; Sun, F. HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 845–853.
9. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
10. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *arXiv*, 2017; arXiv:1707.07998.
11. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv*, 2014; arXiv:1409.0473.
12. Hu, H.; Gu, J.; Zhang, Z.; Dai, J.; Wei, Y. Relation Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
13. Kuo, W.; Hariharan, B.; Malik, J. Deepbox: Learning objectness with convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
14. Pepikj, B.; Stark, M.; Gehler, P.; Schiele, B. Occlusion patterns for object class detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013.
15. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3296–3297.

16. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
17. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
18. Chen, Y.; Li, W.; Sakaridis, C.; Dai, D.; Van Gool, L. Domain Adaptive Faster R-CNN for Object Detection in the Wild. *arXiv*, 2018; arXiv:1803.03243.
19. Wang, X.; Shrivastava, A.; Gupta, A. A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection. *arXiv*, 2017; arXiv:1704.03414.
20. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *arXiv*, 2017; arXiv:1703.06870.
21. Li, J.; Liang, X.; Wei, Y.; Xu, T.; Feng, J.; Yan, S. Perceptual Generative Adversarial Networks for Small Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1951–1959.
22. Koch, C.; Ullman, S. Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry. *Hum. Neurobiol.* **1987**, *4*, 219–227.
23. Itti, L.; Koch, C.; Niebur, E. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 1254–1259. [CrossRef]
24. Bruce, N.D.B.; Tsotsos, J.K. Saliency based on information maximization. In Proceedings of the International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 5–8 December 2005; pp. 155–162.
25. Borji, A. Boosting bottom-up and top-down visual features for saliency estimation. In Proceedings of the IEEE Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 438–445.
26. Wang, X.; Yu, L.; Ren, K.; Tao, G.; Zhang, W.; Yu, Y.; Wang, J. Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 2051–2059.
27. Ren, P.; Chen, Z.; Ren, Z.; Wei, F.; Ma, J.; de Rijke, M. Leveraging Contextual Sentence Relations for Extractive Summarization Using a Neural Attention Model. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 95–104.
28. Seo, S.; Huang, J.; Yang, H.; Liu, Y. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 297–305.
29. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
30. Uijlings, J.R.; van de Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
31. Alexe, B.; Deselaers, T.; Ferrari, V. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2189–2202. [CrossRef] [PubMed]
32. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.B.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014.
33. Noh, H.; Hong, S.; Han, B. Learning deconvolution networkfor semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
34. Badrinarayanan, V.; Handa, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv*, 2015; arXiv:1505.07293.
35. Newell, A.; Yang, K.; Deng, J. Stacked hourglass networks for human pose estimation. *arXiv*, 2016; arXiv:1603.06937.
36. Everingham, M.; van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 2007. Available online: http://www.pascal-network.org/challenges/VOC/voc2007/index.html (accessed on 2 June 2010).
37. Zinkevich, M.; Weimer, M.; Li, L.; Smola, A.J. Parallelized stochastic gradient descent. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 6–9 December 2010.

38. Yang, B.; Yan, J.; Lei, Z.; Li, S.Z. CRAFT Objects from Images. In Proceedings of the IEEE Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 6043–6051.
39. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.

future internet

MDPI

Article

# Bidirectional Recurrent Neural Network Approach for Arabic Named Entity Recognition

**Mohammed N. A. Ali [1], Guanzheng Tan [1,*] and Aamir Hussain [2]**

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China; md.alkhatib@csu.edu.cn
[2] Department of Computer Science, Muhammad Nawaz Shareef University of Agriculture, Multan 60000, Pakistan; aamir.hussain@mnsuam.edu.pk
* Correspondence: tgz@csu.edu.cn

Received: 28 October 2018; Accepted: 10 December 2018; Published: 13 December 2018

**Abstract:** Recurrent neural network (RNN) has achieved remarkable success in sequence labeling tasks with memory requirement. RNN can remember previous information of a sequence and can thus be used to solve natural language processing (NLP) tasks. Named entity recognition (NER) is a common task of NLP and can be considered a classification problem. We propose a bidirectional long short-term memory (LSTM) model for this entity recognition task of the Arabic text. The LSTM network can process sequences and relate to each part of it, which makes it useful for the NER task. Moreover, we use pre-trained word embedding to train the inputs that are fed into the LSTM network. The proposed model is evaluated on a popular dataset called "ANERcorp." Experimental results show that the model with word embedding achieves a high F-score measure of approximately 88.01%.

**Keywords:** Arabic named entity recognition; bidirectional recurrent neural network; GRU; LSTM; natural language processing; word embedding

## 1. Introduction

The Named entity recognition (NER) is important in natural language processing (NLP) tasks used to detect named entities (NEs) in texts and classify them into predefined categories, such as location, person, time, date, and organization [1]. NER is a crucial preprocessing phase in various NLP applications to improve the overall performance. It extracts valuable information from raw data and simplifies downstream tasks, such as text clustering, information retrieval, translation, and question answering [2].

In recent years, Arabic NER (ANER) has become a challenging task and is receiving increasing attention from current researchers due to the limited availability of annotated datasets [3]. Arabic is a Semitic and the standard language spoken in the Arab world. The language is used in the Middle East, the Horn of Africa and North Africa, and it is used in the United Nations as one of the five official languages. In the Arab world, around 360 million people speak Arabic in more than 25 countries [4].

The Arabic NLP has drawn attention in recent years. Several NLP tasks are tricky, such as NER [5], particularly language important features, such as high morphological uncertainty of meaning, writing style, doubtfulness to the meaning of common words/proper noun, and absence of capitalization [2].

The ANER systems are based on either one of two methods: one is based on handcrafted rules, particularly the NERA2.0 system [6], and the other relies on statistical learning such as in [7]. Each method, nonetheless, has its pros and cons. NER systems designed on rule-based primarily depend on manually crafted grammatical rules learned from linguists. Therefore, the maintenance of these systems is time-consuming and laborious, especially when the knowledge and background of the linguists are poor. By contrast, systems based on machine learning (ML) obtain patterns related to the NER task from the training set of samples automatically, thereby not requiring in-depth language-specific knowledge. These ML-based systems are superior to rule-based systems because they are adjustable and easy to update with minimum cost and time, provided a sufficient corpus.

In recent years, neural networks have drawn much attention with various models being proposed. Researchers have combined different semi-supervised learning and deep neural networks (DNNs) to find an optimum solution to the NER task and other chunking tasks [8]. Contrary to ordinary ML methods, deep learning can concurrently learn representations, categorize patterns, and considerably reduce the difficulty in NER tasks. Moreover, current deep learning models generally utilize word embeddings, which allow them to learn similar representations for semantically similar words. However, out-of-vocabulary (OOV) words, which are words that do not have any corresponding representation in the word embedding model, are difficult to handle especially for the Arabic language because of the limitation in resources. These words are also set randomly to a specific value. Therefore, we fully utilize the character representations of a token to label those OOV words. We also introduce the embedding attention layer that works as a gating mechanism that allows the model to dynamically learn what features are important.

ANER is considered a sequence labeling task. Recurrent neural network (RNN) is a natural choice to tackle problems with sequential structure, such as NER, due to their ability to memorize previous values and correlate to other parts of a sequence. RNN surpasses other methods in terms of NER performance and other sequence labeling problems for many languages.

To the best of our knowledge, no work has been done to address the ANER problem using RNN techniques, and most existing works are based on feature engineering and statistical methods. In this work, we propose a new method that has not been investigated in much detail for the Arabic language to date. However, the method has been examined and applied widely in other domains and languages, such as English, and has shown outstanding results.

The model computes the harmonic mean F-score measure for tokens in the dataset. The proposed system has improvements that boost the recognition efficiency and accuracy. The main contribution of our work are listed as follows:

- An ANER system based on bidirectional long short-term memory (Bi-LSTM)/gated recurrent units (GRUs) is proposed by considering the NER problem as a classification problem and without using any manual feature engineering.
- Character embedding is used in addition to word embedding to enhance the model prediction.
- An embedding attention layer that combines word and character embeddings enable the model to create a good word representation.

Our work differs from existing ones for the Arabic language because the proposed model shifts from traditional ML algorithms to neural network algorithms. Moreover, the Bi-LSTM unit uses character and word embeddings as input. A well-recognized dataset called "ANERcorp" is used to evaluate the performance of the proposed system in comparison with other common state of the art systems.

The rest of the paper is organized as follows. Section 2 reviews the related work briefly. Section 3 presents the proposed approach for NER in detail. Section 4 describes the experimental settings. Section 5 discusses the results. Section 6 presents the discussion. Finally, Section 7 elaborates the conclusion.

## 2. Related Work

As per recent systematic review for NER conducted by [9], NER approaches can be categorized into three mains categories rule-based, learning-based and hybrid approaches. Furthermore, another comprehensive survey conducted by W Etaiwi [2], classify the statistical methods of Arabic NER into six main approaches: CRF, NB, HMM, ME, SVM and Neural network. Going more in-depth into deep learning, a survey on ANLP using deep learning techniques by [10]. The author reviewed many kinds of literature on various ANLP tasks and concluded that yet a considerable gap exists in the Arabic NLP compared to that in English NLP.

Basic ML approaches rely on feature engineering, external gazetteers, and chunks. They can achieve excellent accuracy and performance but require a dedicated expert in the domain of knowledge and are time-consuming. Therefore, scholars have begun to consider the artificial neural network (ANN) and DNN methods. These approaches reduce the dependency on feature engineering and are thus less laborious and time-consuming. These methods have been adopted to many NLP problems successfully such as in [11].

The authors in [12] proposed an RNN model for the Chinese Bio-NER that detects two types of predefined annotations, namely, subject and lesion, which are two main parts of symptom entities. A real-world Chinese clinical dataset that includes 12,498 records was used. A priori word information (POS tags) was added for improving the final performance. The final F-scores for the subject and lesion detection reach 90.36% and 90.48% respectively.

In [13], the authors proposed a hybrid NER system for the Russian language that uses a Bi-LSTM-CRF system for various kinds of DNN models experimented starting from vanilla (Bi-LSTM). The models were further supplemented with CRF along with highway networks and added with word embedding. He evaluated all the proposed systems across three datasets: Gareev's, Person-1000, and FactRuEval 2016. He concluded that the quality of predictions is considerably increased with the extension of Bi-LSTM model with CRF and can be further improved by setting the word input to preprocessing with exterior word embedding.

The authors in [14] proposed an improved NER system using deep learning module for Chinese text. Without using any manual feature engineering, the system can detect the word features automatically. He used word embedding with a Bi-LSTM obtained from the outputs of NER to model the substance within a sentence. Along with additional features to the model, the experimental results show that the model achieves an F-score of 0.9247 when trained on a large corpus on Bi-LSTM with word embedding.

The authors in [15] investigated a deep learning method to recognize NEs from Chinese clinical text using the minimal feature engineering approach. Two DNN models were developed: one is for generating the word embedding, and the other is for the main NER task. They evaluated the system on two Chinese clinical datasets: one is an annotated corpus that contains 400 randomly selected admission notes, and the other is unlabeled admission notes that include 36,828 notes, both of which were collected from the EHR database of Peking Union Medical College Hospital in China. The model results indicate that the proposed approach with DNN outperforms the CRF and achieves a high F1-score of 0.9280.

In addition to NER for other languages, such as English, Russian, Chinese, and Hindi, The ANER is an attractive and challenging task and requires a recognition task due to the peculiar and unique characteristics of the Arabic language.

The authors in [16] adopted a new attempt for ANER using ANNs. They used ANN techniques and evaluated the performance of their model with decision trees. Their system consists of three main phases: preprocessing of the data, conversion of Arabic letters to Roman alphabets, and classification of the collected data using a neural network. The relationship between the accuracy of the system and the corpus size was also assessed. The results showed that high accuracy on the same dataset ("ANERCorp") is achieved when an ANN method is adapted to the NER system than when complemented with the decision trees. The experimental results also showed that the accuracy of the system increases proportionally with the enlargement in the size of the corpus.

## 3. Approach

A prototype of bidirectional RNN (B-RNN) is explicitly built on long short-term memory (LSTM)/GRUs. For the entire work, we start with a brief overview of RNN, LSTM, and GRU. Then, we present the bidirectional architecture for the ANER task.

### *3.1. Recurrent Neural Network (RNN)*

RNN is a type of ANN in which the connections between units produce a directed graph along a sequence. This architecture enables the network to demonstrate dynamic temporal behavior for a time sequence. Contrary to feedforward neural networks (FFNNs), RNNs can process sequences of inputs by using their internal state (memory), which makes them appropriate to many NLP tasks [17–19]. LSTM is a type of RNN architecture that is the best among all other existing architectures [20]. RNNs are deep learning systems that stem from the modifications of FFNN with recurrent connections. In a typical neural network, the output of a neuron at time t is calculated as

$$y_i^t = \sigma(W_i x_t + b_i), \tag{1}$$

where $W_i$ is the weight matrix, and $b_i$ is a bias term. In RNN, the calculation of the activation function is modified because the output of the neuron at the time '$t-1$' is fed back into the neuron. Then, the new activation function can be computed as

$$y_i^t = \sigma\left(W_i x_t + U_i y_i^{t-1} + b_i\right). \tag{2}$$

RNNs can remember previous information of a sequence by using the output of the earlier node as recurrent connections while presenting output depending on the former states. This characteristic makes the network useful for the sequence labeling task. The backpropagating error can regularly "strike up" or blast, which yields infeasible convergence. It can also vanish to render the capability of the network to learn long-term dependencies, which are difficult to learn through gradient descent.

### *3.2. Long Short-Term Memory (LSTM)*

LSTM networks are a class of RNNs that are designed efficiently to study long-term dependencies and avert the fading gradient issue. LSTM prevents back-propagating errors from vanishing or exploding. To realize this task, LSTM holds an inner condition that symbolizes the memory cell of the LSTM neuron. The inner condition state is usually augmented by recurrent gates that control the movement of the information over the cell state. These gates are updated and calculated as follows:

$$i_t = \tanh(W_{xi} x_t + W_{hi} h_{t-1}), \tag{3}$$

$$f_t = \sigma\left(W_{xf} x_t + W_{hf} h_{t-1}\right), \tag{4}$$

$$o_t = \sigma(W_{xo} x_t + W_{ho} h_{t-1}), \tag{5}$$

where $i_t$, $f_t$, and $o_t$ represent input, forget, and output gates, respectively. The first two gates decide the input of the last output and the present input in the new cell condition $c_t$. The last gate takes charge of the quantity of the cell state $c_t$, which is exposed as the output. The new $c_t$ and $h_t$ can be computed as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c), \tag{6}$$

$$h_t = o_t \odot \tanh(c_t). \tag{7}$$

The cell state keeps relevant information from the last time steps. The cell state can be updated via the input and forget gates in an additive manner only. This procedure can be regarded as permitting the error to stream back over the cell state unchecked until it gets propagated back to the time step in which the related information is added. This mechanism enables LSTM to study long-term reliance.

### *3.3. Gated Recurrent Units (GRUs)*

GRU, which was first presented in 2014, is a mechanism that uses gates in RNNs. Although GRU is modern, it can be presented as a simplified version of the LSTM. GRU uses different mechanisms for generating the update and updating the cell state. The activation function of the candidates $h_t$ is calculated on the basis of the present input and the last cell state as follows:

$$\hbar_t = \sigma(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1})), \tag{8}$$

where $r_t$ is the reset gate that has the same functional form as that of an update gate, and all of its weights are the same size. Reset gate is multiplied by the previous hidden state value and controls the usage of last cell state while computing the activation input. It can reset the hidden value. The computation of the reset gate is based on the last activation cell $h_{t-1}$ and the present candidate activation.

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1}) \tag{9}$$

The current cell state or activation is a linear combination of the previous cell and candidate activations.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h_t, \tag{10}$$

where $Z_t$ is the update gate that balances the combined amount of the previous and new candidate hidden values to obtain the new hidden value.

$$Z_t = (W_{hz}h_{t-1} + W_{xz}x_t) \tag{11}$$

### *3.4. Proposed Model*

In this section, we discuss our bidirectional model for recognizing NEs. Instead of a feedforward network, Bi-LSTM/GRU network is adapted to obtain bidirectional word arrangement for excellent foresight.

Problem Definition: Given a document D containing a sequence of sentences ($s_1, s_2, \ldots, s_m$) for the input sentence X = ($x_1$, $x_2$, · · ·, $x_n$), Z is identified as the output matrix n × k, where k is the number of labels that represent entities. $Z_{ij}$ refers to the score when the ith token ($x_i$) in the sentence tag$_j$. Y = ($y_1$, $y_2$, $y_3$, · · ·, $y_n$) is the predicted sequence.

First, the word and character embeddings are obtained for the input sentence. Then, we adopt an embedding attention layer that combines the two features to obtain the best word representation. The embedded feature representation is fed into the encoder layer for processing. Finally, the result is obtained from the output layer. Xi refers to the input word, whereas Yi is the predicted tag for the i'th token in the sentence, where $1 \leq i \leq t$. The embedding layer consists of the word vector mapping from tokens to dense n-dimensional vector representations. The model is explained in detail in the following sections. Figure 1 shows the B-RNN setup with the LSTM/GRU network, and the main components are described below.

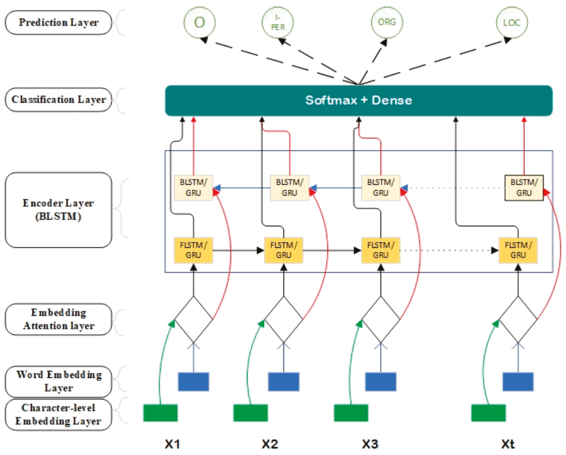


**Figure 1.** Main architecture of the network. Word embedding is provided to a Bi-LSTM/GRU. The forward unit represents the word $X_t$ and its left context, whereas the backward unit represents the word $X_t$ and its right context. Concatenating the two vectors yields a representation of the word $X_t$ and its context, and this embedded representation is fed to the classification layer.

## A. *Embedding Layer*

The work introduced by [21] for word distributed representation has substituted the traditional bag-of-words encoding technique and accomplishes excellent results on many NLP tasks. In distributed embeddings, the model is generalizable because each word gets a map to space. As a result, semantically approximated words can have similar vector representations. However, using word embedding alone as the smallest feature representation unit can result in losing some accurate information. For languages with rich morphology, such as Arabic, we need to capture all morphological and orthographic information. As word embedding encodes semantic and syntactic word relationships, character embedding carries important morphological and shape information. Inspired by this integration as in [22], we acquire the sequence representations from character and word levels.

- Character Embedding Layer

Character sequence representations are helpful for morphologically rich languages and for dealing with the OOV problem for tasks, such as POS tagging and language modeling [23] or dependency parsing [24]. The authors in [25] proposed CharCNN, which is a character-aware neural language model that learns character word representations using CNNs. We follow the same technique for generating the character embedding representation. Details about the implementation can be found in [25].

- Word Embedding

Word embedding refers to the representation of words as vectors in a continuous space, thereby capturing many syntactic and semantic relations among them. We treat the embeddings as fixed constants as this consideration performs better than treating them as learnable model parameters [26]. We adopt pre-trained word embedding, AraVec 2.0 [27], to obtain the fixed word embedding of each token.

## B. *Embedding Attention Layer*

Word embedding treats words as the smallest unit and disregards any morphological resemblances between various words, thereby leading to the OOV problem. By contrast, character embedding can operate over individual characters in each word and can, therefore, be useful for handling OOV. However, research on character embedding is still in the initial phase, and systems that work solely on characters are not superior to those based on words on most tasks [28]. Therefore, character and word embeddings can be integrated to fully utilize their advantages. However, we adopt an embedding attention layer that works as a gate mechanism, can learn similar representations, and allows the model to determine how to consolidate the information for each word. After getting the character feature of each word, we calculate the attention matrix through a two-layer perceptron and combine the two levels of features by a weighted sum as follows:

$$z = \sigma(U_a \tanh(V_a x + W_a m)), \tag{12}$$

$$\widetilde{x} = z{\cdot}x + (1 - z){\cdot}m \tag{13}$$

where $U_a$, $V_a$, and $W_a$ are the weight matrices for computing the attention matrix $z$ and $\sigma()$ is the sigmoid logistic function with values between 0 and 1. $x$ and $m$ are sequence representations of word and character embeddings, respectively. The dimensions of vector $z$ are the same as those of $x$ or $m$ and act as the weight between the two vectors. Accordingly, the model can dynamically determine the amount of information to use from each of the embeddings (character or word).

### *C. Bidirectional Recurrent Neural Networks (B-RNNs)*

Despite its simplicity, B-RNN is a powerful way to improve the neural network ability to learn, especially for NLP problems. The need for B-RNN helps us consider it in the context of NLP tasks, especially NER in which we try to extract NEs, such as people, location, organization, date, and time.

We consider the three sentences below:

**سبأ** مملكة عربية يمنية قديمة "Saba is an ancient Yemeni kingdom."

**سبأ** للصناعات الكيماوية المتحدة المحدودة "Saba United Chemical Industries Ltd."

**سبأ** بنت مبخوت الشهراني زعيم أكبر القبائل العربية "Saba, the daughter of Mbkhout Shahrani leader of the largest Arab tribes."

The word **"- سبأSaba"** in the three sentences will be tagged with something else in each sentence. This word represents a location in the first sentence, a company name in the second sentence, and a proper noun in the third sentence. The problem is that the word **"سبأ"** appears at the beginning of each sentence. This condition indicates that the RNN network will not detect any other word in the sentence before it has the opportunity to make a prediction on the word **"سبأ"**, resulting in a possible incorrect prediction. B-RNN solves this problem by traversing the sequence in both directions. The backward RNN will calculate $\overleftarrow{h_T}$ in reverse direction, starting from $h_T$ and then going backward until $h_1$ for ease of prediction of the right label for the given NE. The idea of going backward helps accurately mark the NEs for the word **"سبأ"** that appears at the beginning of the sentence.

The bidirectional unit contains two LSTM/GRU series: one is propagating in the forward direction, and the other is propagating toward the back direction. We concatenate the outcome from the two series to establish a joint representation of the word and its context.

$$h_t = [\overrightarrow{h_T}, \overleftarrow{h_1}], \tag{14}$$

where $\overrightarrow{h_T}$ is the last word in the forward chain and $\overleftarrow{h_1}$ is the last word in the backward chain.

### *D. Prediction Layer*

The joined vectors from the B-LSTM/GRU network are fed into a classification layer with a feedforward neuron. Furthermore, a SoftMax function is used to normalize the output, and it is given by the following equation:

$$output = softmax\ (h_t). \tag{15}$$

For each tag type j, the probability of similar outputs can be calculated as follows:

$$P(l_t = j|u_t) = \frac{\exp(u_t W_j)}{\sum_{k=1}^{K} \exp(u_t W_k)}, \tag{16}$$

where $l_t$ and $u_t$ are the tags or labels and the concatenated vector for each time step $t$. The highest possible tag at each word position is chosen. The entire network is trained via backpropagation. The embedding vectors are updated on the basis of back-propagating errors as well.

## 4. Experiment

A series of extensive experimentation was conducted to validate the methodology. The datasets used and the experimental setup were explained thoroughly.

### *4.1. Datasets*

To train and test our ANER system, we evaluated the system with "ANERCorp," which is a dataset created by Benajiba from several online resources [29]. The ANERCorp dataset is a manually annotated corpus that is freely available for research purposes. Two corpora were used: training and testing. One person annotated the corpus to guarantee the coherence of the annotation, and it had 4901 sentences with 150,286 tokens. Each line contained a single token for easy parsing. Each word in this dataset was tagged with one of the following: person, location, company, and others. ANERcorp was annotated into eight classes: B-PERS, beginning of the person's name; I-PERS, inside of the person's name; B-LOC, beginning of the location's name; B-ORG, beginning of the organization's name; I-ORG, inside of the organization's name; B-MISC, beginning of the miscellaneous word; I-MISC, inside of the miscellaneous word; O, the word that is not an NE but refers to other NEs. The dataset distribution was as follows: 39% for person, 30.4% for location, 20.6% for organization, and the remaining 10% for miscellaneous.

### *4.2. Baseline*

Many approaches tackle the ANER problem. We selected some of the works that have been carried out previously and compared them with our work by using the same dataset and evaluation metrics. The following works were selected as baselines:

- The work using a minimally supervised approach for ANER proposed in [30]
- The approach using CRFs by Yassine Benajiba in [31]
- The ANN approach using a neural network technique by Naji and Nazlia in [16]

### *4.3. Setting*

An NVIDIA GeForce GTX1080Ti (12 GB and Intel i7-6800K 3.4GHZx12 Processor with 32 GB RAM) was used to train the model. It was built on Ubuntu and implemented in the Keras environment. For each token, the model was trained to predict either one of the eight appropriate labels described in Section 4.1. The embedding dimension was fixed to 100, and the size of the hidden state was aligned to 200. The combination of forward and backward LSTM provided a dimension of 400 Tanh, which was used as the hidden activation function. Its output was fed into a Softmax output layer to produce probabilities for each of the eight tags. Categorical cross-entropy was used as the objective function, and L2-regularization component was added to the cost function for output tuning. For the over-fitting problem, 50% dropout was used as an additional measure to control the inputs to the LSTM network and the Softmax layer. AdaGrad was used to optimize the network cost. The batch sizes were set to 128. A total of 30 epochs were used to train each network. We set the maximum sequence length to 100 to ensure the same length of all the sequences. Sequence greater than this length would be truncated and sequence less than this length would be padded with zeros to obtain the same length.

### 4.4. Evaluation

The efficiency of the system was assessed by the general measures, namely, precision ($P$), recall ($R$), and F-measure score ($F$), because they are standard for NER.

$$P = \frac{X}{Y}, \tag{17}$$

$$R = \frac{X}{Z}, \tag{18}$$

$$F - measure(F) = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{19}$$

where $X$ is the total number of correctly extracted entities, $Y$ is the entire number of recognized entities, and $Z$ is the total number of correct entities.

## 5. Results

We run our experiments on the ANERCorp dataset described above. To evaluate the performance of our Bi-LSTM/GRU model, we run each model 5 times and take the average of each score value. The experimental results are summarized in Table 1. RNN can handle the sequence labeling problem efficiently without the need for additional information, such as Chunks or Gazetteers, which are essential especially for the NER task.

**Table 1.** Results obtained by our model with different architectures.

| Model | *P* | *R* | *F*1 |
|---|---|---|---|
| LSTM | 80.49 | 79.71 | 80.07 |
| GRU | 75.46 | 76.86 | 76.13 |
| BLSTM | 86.85 | 86.01 | 86.42 |
| BGRU | 85.70 | 83.96 | 84.81 |
| BLSTM + char | 88.49 | 87.57 | 88.01 |
| BGRU + char | 87.73 | 86.51 | 87.12 |

The experimental results show that our proposed model with embedding attention layer performs considerably better than the other neural NE recognition and previous baseline systems. The best F-scores of our model are 88.01% and 87.12% for BLSTM and BGRU, respectively.

## 6. Discussion

### 6.1. Comparison of Different Architectures

To evaluate the performance of our Bi-LSTM/GRU model and show the effect of different architectures (i.e., the bidirectionality and the impact of character embedding), we first run our experiment on the unidirectional LSTM/GRU. Then, we run the Bi-LSTM/GRU. Finally, we add the character embedding. As observed from the results above, the model achieves 80.07% and 76.13% for LSTM and GRU, respectively. The model obtains slight improvement in bidirectionality and achieves the best results of 88.01% and 87.12% for BLSTM and BGRU, respectively, when the character embedding is added to it. Notably, the performance of B-LSTM and B-GRU is nearly similar and slightly better than that of LSTM. Moreover, RNN with word embedding can perform effectively in the ANER task without the need for manual feature engineering.

### 6.2. Comparison with Other Models

The performance of the proposed model is compared with those of other existing state of the arts. Table 2 shows the comparison results obtained by our model against those of other systems. Our proposed model, B-LSTM/GRU, achieves the best result regarding the metrics used for evaluation on the "ANERCorp" dataset. This performance emphasizes that using RNN leads to excellent performance and improves the labeling task. Specifically, the efficiency results obtained on the NER task are comparable to those of previous methods.

**Table 2.** Comparison of F-score performance measure of the proposed models concerning the baseline systems on ANERCorp.

| Model | *P*% | *R*% | *F*% |
|---|---|---|---|
| Maha Althobaiti [30] | 84.94 | 52.68 | 63.22 |
| Benajiba (ANERsys 2.0) [31] | 65.33 | 58.60 | 61.73 |
| Nazlia Omar [16] | 65.03 | 62.33 | 57.47 |
| F Dahan (HMM) [7] | 77 | 73 | 75.2 |
| **OUR B-LSTM** | 88.49 | 87.57 | **88.01** |
| **OUR B-GRU** | 87.73 | 86.51 | **87.12** |

### *6.3. Effect of Character Embedding*

Apart from the effect of word embedding, which is a powerful tool to learn word representation and can perform excellently on various NLP tasks, character embedding enhances the performance of the model by handling the OOV problem and providing an accurate representation for words that do not have corresponding word embeddings.

## 7. Conclusions

Arabic is a challenging language because of its high morphological ambiguity, writing style, and absence of capitalization. Thus, any NLP task for this language requires a large number of feature engineering and preprocessing steps. In this study, we experiment with a B-RNN with LSTM/GRU for the ANER task. Without using any feature engineering or additional preprocessing, we tackle the problem of NER for the Arabic text. We find that deep learning-based approaches, especially LSTM network, are useful for identifying Arabic NEs and can efficiently outperform many other methods based on manually engineered features or rule systems. The incorporation of pre-trained word embedding enables the system to obtain considerable improvements in the recognition task and achieve excellent results in F-score measures, where we achieved a high F-score measure of approximately 88.01% and 87.12% for Bi-LSTM and Bi-GRU respectively.

**Author Contributions:** M.N.A.A. wrote the paper with support from A.H. A.H carried out the experiments with the help of M.N.A.A. G.T. supervised the whole project. All the authors revised the manuscript.

**Funding:** This work was partially funded by the National Natural Science Foundation of China (Project No. 61403422).

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

## References

1. Shaalan, K.; Oudah, M. A hybrid approach to Arabic named entity recognition. *J. Inf. Sci.* **2014**, *40*, 67–87. [CrossRef]
2. Etaiwi, W.; Awajan, A.; Suleiman, D. Statistical Arabic Name Entity Recognition Approaches: A Survey. *Procedia Comput. Sci.* **2017**, *113*, 57–64. [CrossRef]
3. Zirikly, A.; Diab, M. Named entity recognition for arabic social media. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, Berlin, Germany, 12 August 2016; pp. 176–185.
4. Nydell, M.K. *Understanding Arabs: A Guide for Modern Times*; Intercultural Press: Boston, MA, USA, 2018.
5. Shaalan, K.; Raza, H. NERA: Named entity recognition for Arabic. *J. Assoc. Inf. Sci. Technol.* **2009**, *60*, 1652–1663. [CrossRef]
6. Oudah, M.; Shaalan, K. NERA 2.0: Improving coverage and performance of rule-based named entity recognition for Arabic. *Nat. Lang. Eng.* **2017**, *23*, 441–472. [CrossRef]

7. Dahan, F.; Touir, A.; Mathkour, H. First Order Hidden Markov Model for Automatic Arabic Name Entity Recognition. *Int. J. Comput. Appl.* **2015**, *123*, 37–40. [CrossRef]
8. Tomas, M. Statistical Language Models Based on Neural Networks. Available online: http://www.fit.vutbr.cz/~{}imikolov/rnnlm/google.pdf (accessed on 9 December 2018).
9. Goyal, A.; Gupta, V.; Kumar, M. Recent Named Entity Recognition and Classification techniques: A systematic review. *Comput. Sci. Rev.* **2018**, *29*, 21–43. [CrossRef]
10. Al-Ayyoub, M.; Nuseir, A.; Alsmearat, K.; Jararweh, Y.; Gupta, B. Deep learning for Arabic NLP: A survey. *J. Comput. Sci.* **2018**, *26*, 522–531. [CrossRef]
11. Awad, D.; Sabty, C.; Elmahdy, M.; Abdennadher, S. Arabic Name Entity Recognition Using Deep Learning. In Proceedings of the International Conference on Statistical Language and Speech Processing, Mons, Belgium, 15–16 October 2018; pp. 105–116.
12. Li, J.; Zhao, S.; Yang, J.; Huang, Z.; Liu, B.; Chen, S.H.; Pan, H.; Wang, Q. WCP-RNN: A novel RNN-based approach for Bio-NER in Chinese EMRs: Paper ID: FC_17_25. *J. Supercomput.* **2018**. [CrossRef]
13. Le, T.A.; Arkhipov, M.Y.; Burtsev, M.S. Application of a hybrid Bi-LSTM-CRF Model to the task of Russian named entity recognition. *Commun. Comput. Inf. Sci.* **2018**, *789*, 91–103.
14. Ouyang, L.; Tian, Y.; Tang, H.; Zhang, B. Chinese Named Entity Recognition Based on B-LSTM Neural Network with Additional Features. In Proceedings of the International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, Guangzhou, China, 12–15 December 2017; pp. 269–279.
15. Wu, Y.; Jiang, M.; Lei, J.; Xu, H. Named Entity Recognition in Chinese Clinical Text Using Deep Neural Network. *Stud. Health Technol. Inform.* **2015**, *216*, 624–628. [PubMed]
16. Mohammed, N.F.; Omar, N. Arabic named entity recognition using artificial neural network. *J. Comput. Sci.* **2012**, *8*, 1285–1293.
17. Yousfi, S.; Berrani, S.-A.; Garcia, C. Contribution of recurrent connectionist language models in improving LSTM-based Arabic text recognition in videos. *Pattern Recognit.* **2017**, *64*, 245–254. [CrossRef]
18. Baly, R.; Hajj, H.; Habash, N.; Shaban, K.B.; El-Hajj, W. A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in arabic. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2017**, *16*, 23. [CrossRef]
19. Chherawala, Y.; Roy, P.P.; Cheriet, M. Feature set evaluation for offline handwriting recognition systems: Application to the recurrent neural network model. *IEEE Trans. Cybern.* **2016**, *46*, 2825–2836. [CrossRef] [PubMed]
20. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An empirical exploration of recurrent network architectures. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2342–2350.
21. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
22. Li, F.; Zhang, M.; Fu, G.; Ji, D. A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinform.* **2017**, *18*, 198. [CrossRef] [PubMed]
23. Ling, W.; Luís, T.; Marujo, L.; Astudillo, R.F.; Amir, S.; Dyer, C.; Black, A.W.; Trancoso, I. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv* **2015**, arXiv:1508.02096.
24. Ballesteros, M.; Dyer, C.; Smith, N.A. Improved transition-based parsing by modeling characters instead of words with LSTMs. *arXiv* **2015**, arXiv:1508.00657.
25. Kim, Y.; Jernite, Y.; Sontag, D.; Rush, A.M. Character-Aware Neural Language Models. In Proceedings of the AAAI, Phoenix, AZ, USA, 12–17 February 2016; pp. 2741–2749.
26. Cocos, A.; Fiks, A.G.; Masino, A.J. Deep learning for pharmacovigilance: Recurrent neural network architectures for labeling adverse drug reactions in Twitter posts. *J. Am. Med. Inform. Assoc.* **2017**, *24*, 813–821. [CrossRef] [PubMed]
27. Soliman, A.B.; Eissa, K.; El-Beltagy, S.R. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Comput. Sci.* **2017**, *117*, 256–265. [CrossRef]
28. Rei, M.; Crichton, G.K.O.; Pyysalo, S. Attending to characters in neural sequence labeling models. *arXiv* **2016**, arXiv:1611.04361.

29. Benajiba, Y.; Rosso, P.; Benedíruiz, J.M. Anersys: An arabic named entity recognition system based on maximum entropy. In Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico, 18–24 February 2007.
30. Kruschwitz, U.; Poesio, M. Combining minimally-supervised methods for arabic named entity recognition. *Trans. Assoc. Comput. Linguist.* **2015**, *3*, 243–255.
31. Benajiba, Y.; Rosso, P. Arabic Named Entity Recognition using Conditional Random Fields. *Proc. Work. HLT NLP Arab. World LREC* **2008**, *8*, 143–153.

*Article*

# Neurologist Standard Classification of Facial Nerve Paralysis with Deep Neural Networks

**Anping Song *, Zuoyu Wu, Xuehai Ding, Qian Hu and Xinyi Di**

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; zzzuo@i.shu.edu.cn (Z.W.); dinghai@shu.edu.cn (X.D.); ileven@shu.edu.cn (Q.H.); jsbluecat@shu.edu.cn (X.D.)

* Correspondence: apsong@shu.edu.cn; Tel.: +86-136-5167-5579; Fax: +86-6613-5550

Received: 27 October 2018; Accepted: 15 November 2018; Published: 16 November 2018

**Abstract:** Facial nerve paralysis (FNP) is the most common form of facial nerve damage, which leads to significant physical pain and abnormal function in patients. Traditional FNP detection methods are based on visual diagnosis, which relies solely on the physician's assessment. The use of objective measurements can reduce the frequency of errors which are caused by subjective methods. Hence, a fast, accurate, and objective computer method for FNP classification is proposed that uses a single Convolutional neural network (CNN), trained end-to-end directly from images, with only pixels and disease labels as inputs. We trained the CNN using a dataset of 1049 clinical images and divided the dataset into 7 categories based on classification standards with the help of neurologists. We tested its performance against the neurologists' ground truth, and our results matched the neurologists' level with 97.5% accuracy.

**Keywords:** facial image analysis; facial nerve paralysis; deep convolutional neural networks; image classification

## 1. Introduction

Facial nerve paralysis (FNP) is one of the most common facial neurological dysfunctions, in which the facial muscles appear to droop or weaken. Such cases are often accompanied by the patient having difficulty chewing, speaking, swallowing, and expressing emotions. Furthermore, the face is a crucial component of beauty, expression, and sexual attraction. As the treatment of FNP requires an assessment to plan for interventions aimed at the recovery of normal facial motion, the accurate assessment of the extent of FNP is a vital concern. However, existing methods for FNP diagnosis are inaccurate and nonquantitative. In this paper, we focus on computer-aided FNP grading and analysis systems to ensure the accuracy of the diagnosis.

Facial nerve paralysis grading systems have long been an important clinical assessment tool; examples include the House–Brackmann system (HB) [1], the Toronto facial grading system [2,3], the Sunnybrook grading system [4], and the Facial Nerve Grading System 2.0 (FNGS2.0) [5]. However, these methods are highly dependent on the clinician's subjective observations and judgment, which makes them problematic with regard to integration, feasibility, accuracy, reliability, and reproducibility of results.

Computer-aided analysis systems have been widely employed for FNP diagnosis. Many such systems have been created to measure facial movement dysfunction and its level of severity, and rely on the use of objective measurements to reduce errors brought about through the use of subjective methods.

Anguraj et al. [6] utilized Canny edge detection to locate a mouth edge and eyebrow, and Sobel edge detection to find the edges of the lateral canthus and the infraorbital region. Nevertheless, these edge detection techniques are very vulnerable to noise. Neely [7–9] and Mcgrenary [10] used a dynamic video image analysis system which analyzed patients' clinical images to assess FNP. They used very simple neural networks on FNP, which validated the technology's potential. Although their results were consistent with the HB scoring system, they had a very small dataset and their system's image processing was computationally intensive. He et al. [11] used optical-flow tracking and texture analysis to solve the problem using image processing to capture the asymmetry of facial movements by analyzing the patients' video data, but this is computationally intensive. Wachtman et al. [12] measured asymmetry using static images, but their method is sensitive to extrinsic facial asymmetry caused by orientation, illumination, and shadows.

For our method, a new FNP classification standard was established based on FNGS2.0 and asymmetry. FNGS2.0 is a widely used assessment system which has been found to be highly consistent with clinical observations and judgment, achieving 84.8% agreement with neurologist assessments [13].

Using deep learning to detect facial landmarks in our previous method has shown promising results. Deep convolutional neural networks (DCNNs) [14] show potential for general and highly variable tasks on image classification [15–19]. Deep learning algorithms have recently been shown to exceed human performance in visual tasks like playing Atari games [20] and recognizing objects [16]. In this paper, we outline the development of a CNN that matches neurologist performance for human facial nerve paralysis using only image-based classification.

GoogleNet Inception v3 CNN architecture [18] was pretrained on approximately 1.28 million images (1000 object categories) for the 2014 ImageNet Large Scale Visual Recognition Challenge [16]. Sun et al. [21] proposed an effective means for learning high-level overcomplete features with deep neural networks called DeepID CNN, which classified faces according to their identities.

At the same time, DCNNs have had many outstanding achievements as diagnostic aids. Rajpurkar et al. [22] developed a 34-layer CNN which exceeds the performance of board-certified cardiologists in detecting a wide range of heart arrhythmias from electrocardiograms recorded using a single-lead wearable monitor. Hoochang et al. [23] used a CNN combined with transfer learning on computer-aided detection. They studied two specific computer-aided detection (CADe) problems, namely thoraco-abdominal lymph node (LN) detection and interstitial lung disease (ILD) classification. They achieved state-of-the-art performance on mediastinal LN detection and reported the first fivefold cross-validation classification results on predicting axial CT slices with ILD categories. Esteva et al. [15] used a pretrained GoogleNet Inception v3 CNN on skin cancer classification, which matched the performance of dermatologists in three key diagnostic tasks: melanoma classification, melanoma classification using dermoscopy, and carcinoma classification. Sajid et al. [24] used a CNN model to classify facial images affected by FNP into the five distinct degrees established by House and Brackmann. Sajid used a Generative Adversarial Network (GAN) to prevent overfitting in training. His research demonstrates the potential of deep learning on FNP classification, even though his final classification accuracy results were not very good (89.10–96.90%, depending on the class). Meanwhile, they used a traditional grading standard to directly label the data which may cause erroneous labeling. They also used four complicated image preprocessing steps, which cannot be automated and which require a lot of time and effort during the clinical diagnosis phase for data labeling.

In the process of realizing a reliable computer-aided analysis system, we also proposed a method for FNP quantitative assessment [25]. We used a DCNN to obtain facial features, then we used asymmetry algorithm to calculate FNP degree. In this work, we validated the effectiveness of DCNN. However, there is currently no work related to the hierarchical classification of FNP using DCNN.

The difficulty of FNP classification lies first and foremost in image classification, followed by face recognition. To design a responsive and accurate CNN for FNP classification, we combined a GoogleNet Inception v3 CNN and a DeepID CNN to design a new CNN called Inception-DeepID-FNP (IDFNP) CNN. As it is difficult to obtain a large enough training dataset, direct training of our model would cause overfitting results, so we need to use transfer learning methods [26] to eliminate overfitting, as given the amount of expected data available, this was considered to be the optimal choice. We trained the IDFNP CNN by training on ImageNet with no final classification layer and then retrained it using our dataset. This method is optimal given the amount of data available.

Compared with other classification methods, we set up our own dataset classification standards. We used deep learning to directly classify FNP, which allows each FNP image to be processed more quickly, has more accurate classification, and has lower image quality requirements. In order to improve the liability and accuracy of our labeling results, we used a triple-check method to complete the labeling of the image dataset. At the same time, we combined image classification with face recognition.

Using the proposed system, clinicians can quickly obtain the degree of facial paralysis under different movements and make a prediagnosis of facial nerve condition, which can then be used as a reference for final diagnosis. At the same time, we also developed a mobile phone application that enables patients to perform self-evaluations, which can help them avoid unnecessary visits to hospitals.

The remainder of this paper is structured as follows.

The proposed methodology is presented in Section 2. The experiments and results are given in Section 3. The results and related discussion are presented in Section 4. The conclusions about this study are given in Section 5.

## 2. Materials and Methods

### *2.1. Data Sources*

We used two types of data sources, a fixed camera in a hospital and a mobile application.

#### 2.1.1. Hospital Camera

In order to establish a novel method for quantitative FNP assessment, we prepared a fixed scene in the Department of Rehabilitation at the Shanghai Tenth People's Hospital in order to obtain FNP images with the neurologists' help. We captured front-view facial images of the patients using reasonable illumination to reduce any adverse illumination effects. The procedure for obtaining the images was standardized; photography was executed while the participant was seated in a chair, and a reference background was placed behind. The camera was mounted on a sturdy tripod at a distance of 1.5 m from the participant, and the latter was instructed to look directly at the camera with their chin raised. Then, digital images were acquired as each participant performed each of the different movements.

#### 2.1.2. Mobile Application

For the purposes of the present study, we developed a mobile application for both iPhone and Android devices, with the end-goal being that patients would be able to obtain an automated preassessment of the extent of their FNP using their mobile phone camera. Participants were asked to download the application, which used the phone's camera and suitable prompts to obtain the relevant images of the participant.

### *2.2. Dataset*

Our dataset came from a combination of an FNP dataset and a normal dataset. The FNP dataset came from clinical images from the Department of Rehabilitation at the Shanghai Tenth People's Hospital. The FNP dataset was composed of 377 male images and 483 female images, of which 136 were of patients less than 40 years old, 302 were middle-aged (between 40 and 65 years old), and 422 were elderly (greater than 65 years old). The normal dataset was composed of recovered patients, volunteers to our research group, and healthy neurologists from the hospital's Department of Rehabilitation. The normal dataset was composed of 86 male normal images and 103 female images, of which 38 were less than 40 years old, 82 were between 40 and 65 years old, and 69 were elderly (Table 1). Our dataset covers patients of all ages and genders, while patient data are relatively evenly distributed.

**Table 1.** Dataset Distribution.

| Dataset | Young | Middle-Aged | Elderly | Male | Female | Total |
|---|---|---|---|---|---|---|
| FNP images | 136 | 302 | 422 | 377 | 483 | 860 |
| Normal images | 38 | 82 | 69 | 86 | 103 | 189 |
| Total | 174 | 384 | 491 | 463 | 586 | 1049 |

Figures 1 and 2, respectively, show example facial images of the control and the patient groups taken as each group was performing seven facial movement types: at rest, eyes closed, eyebrows raised, cheeks puffed, grinning, nose wrinkled, and whistling. Table 2 contains a description of each movement. These images were used for our model's training.

**Table 2.** Taxonomy movements table.

| Notation | Movement | Affected Facial Muscle |
|---|---|---|
| MV0 | At rest | All facial muscles |
| MV1 | Eyes closed | Orbicularis oculi muscle |
| MV2 | Eyebrows raised | Orbicularis oculi muscle, frontalis muscle |
| MV3 | Cheeks puffed | Orbicularis oculi muscle, buccinator muscle, zygomatic muscle |
| MV4 | Grinning | Orbicularis oris muscle |
| MV5 | Nose wrinkled | Nasalis muscle |
| MV6 | Whistling | Orbicularis oris muscle |

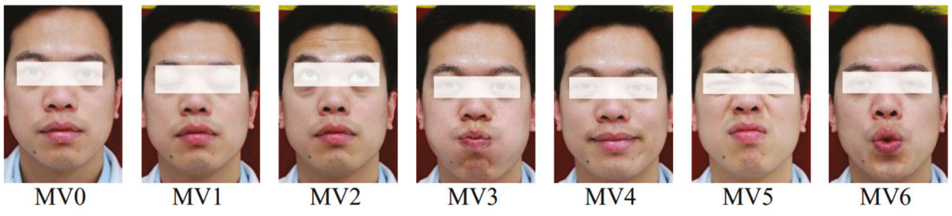


**Figure 1.** Facial images of the control subjects during seven movement types.

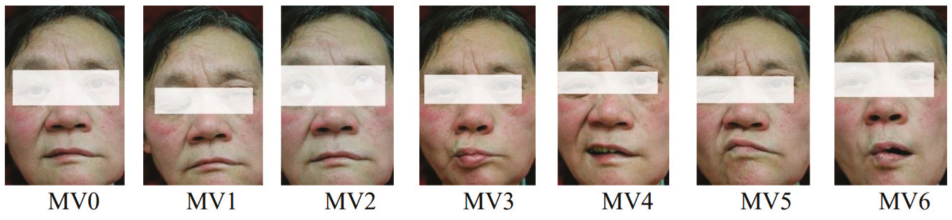


**Figure 2.** Facial images of patients with paralysis during the seven movement types.

## 2.3. *Taxonomy*

### 2.3.1. Classification Standard

Since FNP causes barriers to the movement of facial muscles, we can evaluate the degree of FNP by calculating the asymmetry of facial features for different facial movements. This method was chosen because simultaneous bilateral FNP is highly improbable. Our method is based on facial image analysis. Considering our dataset consists of FNP images and not video, in order to reduce subjective factors and the difficulty of diagnosis, the new classification standard divides the dataset into seven categories. These are: normal, left mild dysfunction, left moderate dysfunction, left severe dysfunction, right mild dysfunction, right moderate dysfunction, and right severe dysfunction (Table 3).

**Table 3.** Taxonomy characteristics.

| Taxonomy | Symbol | Characteristics |
|---|---|---|
| normal | N | Normal function in all facial nerve areas. |
| left mild dysfunction | L1 | Slight muscular weakness observed on examination of the side of the face. Facial image appears symmetrical and with tones in MV0, eye area exhibits mild asymmetry in MV1, eyebrow area exhibits mild asymmetry in MV2, cheek area exhibits mild asymmetry in MV3, mouth area exhibits mild asymmetry in MV4 and MV6, nose and cheek areas exhibit asymmetry in MV5. |
| right mild dysfunction | R1 | |
| left moderate dysfunction | L2 | In the side of the face, there is a clear difference between the two hemifaces but this is not total asymmetry. Nonserious disordered movement may be observed. Facial image exhibits mild asymmetry in MV0, eye area exhibits moderate asymmetry in MV1, eyebrow area exhibits moderate asymmetry in MV2, cheek area exhibits moderate asymmetry in MV3, mouth area exhibits moderate asymmetry in MV4 and MV6, nose and cheek areas exhibit moderate asymmetry in MV5. |
| right moderate dysfunction | R2 | |
| left severe dysfunction | L3 | In the side of the face, there is clear weakness and total asymmetry, with hardly any observable mobility. Facial image exhibits severe asymmetry in MV0, eye area exhibits severe asymmetry in MV1, eyebrow area exhibits severe asymmetry in MV2, cheek area exhibits severe asymmetry in MV3, mouth area exhibits severe asymmetry in MV4 and MV6, nose and cheek areas exhibit severe asymmetry in MV5. |
| right severe dysfunction | R3 | |

### 2.3.2. Frequencies in Dataset Taxonomy

Our taxonomy represents seven different classes of FNP and their frequency for the study sample is given in Table 4. This aspect of the taxonomy is useful for generating training classes that are well suited for machine learning classifiers. We obtained 664 images from the hospital camera and 385 images from the application.

**Table 4.** Frequencies in dataset taxonomy.

| **Taxonomy** | **N** | **L1** | **L2** | **L3** |
|---|---|---|---|---|
| Frequency | 189 | 133 | 161 | 146 |
| **Taxonomy** | **R1** | **R2** | **R3** | |
| Frequency | 129 | 151 | 140 | |

### 2.3.3. Labeling

In order to objectively divide image database into those seven categories, we used a triple-check method to complete the labeling of the image dataset.

To start with, neurologists labeled images into seven different categories twice, and only coinciding labels were retained for subsequent steps. This was the first check in the process.

Then, we measured the degree of bilateral face FNP difference using asymmetry [25]. In order to measure the asymmetry of patients during different facial movements, we assessed eye asymmetry (EAs), eyebrow asymmetry (EBAs), nose asymmetry (NAs), mouth asymmetry (MAs), mouth angle (MAn), nose angle (NAn), and eyebrow angle (EbAn). We quantified this assessment using two variables, regional asymmetry (RgAs) and angular asymmetry (AnAs), which were calculated using the following equation:

$$RgAs = EAs + EBAs + NAs + MAs \tag{1}$$

$$AnAs = MAn + NAn + EbAn \tag{2}$$

Based on the results of the first check, we obtained the range of $RgAs$ and $AnAs$ for every movement type in the same manner for the seven categories.

Since the results of this work are not accurate enough, the work on the classification of the face can only be used as a reference, so we still need to optimize the results to ensure the accuracy of the labeling. We compared the results of the asymmetrical algorithm with the first-check results as reference and kept the coinciding results to obtain the second-check result. Neurologists will take the results of the asymmetrical algorithm as reference to analyze the different part above. Finally, neurologists will obtain the final classification results for the third check.

Using this approach, the results of the first check reached 97% agreement, and for the second check, we achieved 93% agreement.

#### 2.3.4. Data Preparation

Since our data came from two different sources, data transformation was the first step of our method. The biggest difference between the two data sources were the environmental factors. The FNP images taken on the mobile phone application suffered from problems with face angle and image size. We therefore preprocessed the images to obtain a standardized format of the face image. In order to eliminate the influence of environmental factors, we cropped every image. To make them compatible with the IDFNP CNN architecture, we resized each image to $299 \times 299 \times 3$ pixels, which were used as the input to IDFNP. However, because the image size was fixed at $299 \times 299$, and image cropping may have resulted in loss of facial nerve information, cropping was adjusted according to the specific facial movement being captured. In order to retain as much facial nerve muscle information as possible, cropping retained all parts of the muscle for a specific movement. Pictures were cropped automatically and the results were visually inspected and, if necessary, corrected manually to ensure that no useful information was discarded.

Blurry images and distant images were removed from the test and validation sets, but were still used for training. While this is useful training data, extensive care was taken to ensure that these sets were not split between the training and validation sets. No overlap (that is, same lesion, multiple viewpoints) existed between the test sets and the training/validation data.

Based on the above principles, the 1049 images selected after filtering were randomly and evenly divided using a 7:2:1 ratio for the training, verification, and test sets, respectively. The training set batch size was 60, the cross-validated batch size was 100, and for k-fold cross-validation we used $k = 10$.

### *2.4. Model Architecture*

The difficulty of FNP classification lies first and foremost in image classification, followed by face recognition. Inception v3 CNN [18] shows great performance on image classification and won first prize during the 2015 ImageNet Large Scale Visual Recognition Challenge [16]. At the same time, DeepID CNN [21] is the top model in the field of face recognition. In order to design a model for FNP classification, we combined the best image classification CNN model and the best face recognition CNN model for the learning task. In order to combine GoogleNet Inception v3 CNN and DeepID CNN, and thereby create IDFNP CNN, we must identify their essential components and utilize them.

The complete model is based on the Inception-v3 architecture. Apart from the essential components of Inception-v3 and DeepID, IDFNP used a concat layer to concatenate the parameters of the two parts. After the above, the FNP grade classification task is performed by the softmax layer.

The network's high-level architecture is shown in Figure 3.

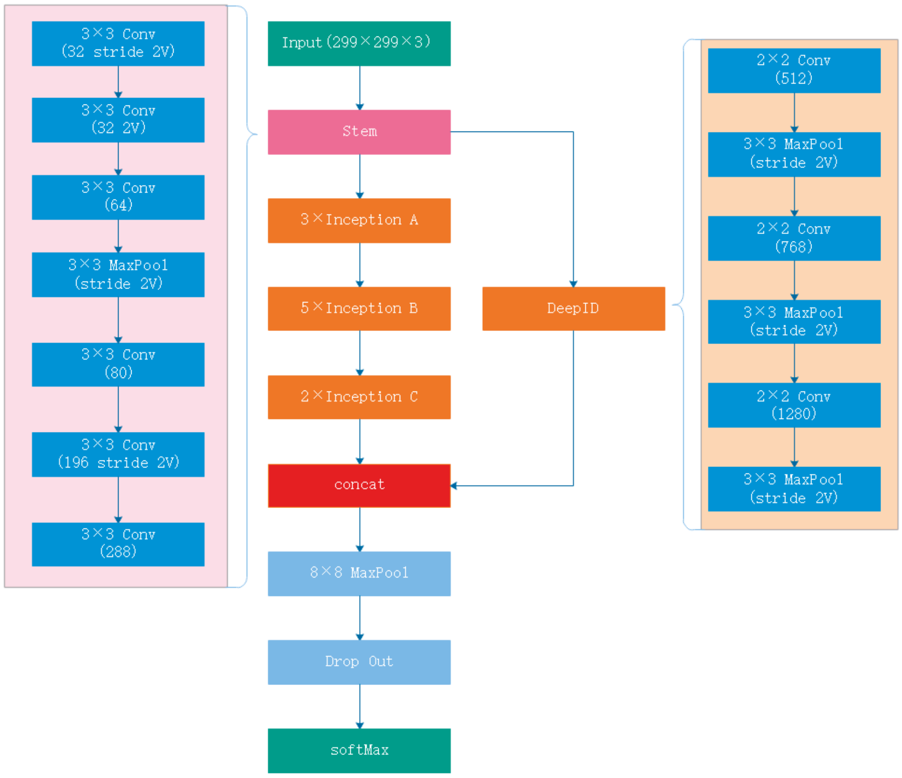


**Figure 3.** Inception-DeepID-FNP CNN.

Because FNP classification counts as image classification, putting the DeepID CNN part into GoogleNet Inception v3 CNN was our strategy of choice. Since the DeepID CNN has much fewer characteristics than GoogleNet Inception v3 CNN, we fine-tuned the parameters across multiple layers in order to enhance the human face component.

## 2.5. Training Algorithm

As it is difficult to obtain a large enough training dataset, direct training of our model would cause overfitting results, so we needed to use migration study methods to eliminate overfitting. Given the amount of expected data available, transfer learning was considered to be the optimal choice.

The ImageNet Challenge Database is a 1000 object class (1.28 million images) image database. Pretraining the model on ImageNet Challenge Database will increase the model's sensitivity to image classification. FNP image classification is based on the details and characteristics of facial muscles, while ImageNet classification is based on the details and characteristics of the classification for which it is trained. The data distribution of the FNP database and ImageNet Challenge Database are similar and, in this case, we transferred the model from a source domain (pretrained model) to a target domain (final model).

The IDFNP CNN is based on Inception-v3 CNN, which has very good performance in the ImageNet Challenge Database. Therefore, we pretrained the IDFNP CNN on the ImageNet Challenge Database and achieved a 93.33% classification accuracy, ranking top-five compared with other CNNs. We then removed the final classification layer from the network, retrained it with our own dataset, and leveraged the natural-image features already learned by the ImageNet pretrained network.

The classification task is performed by the softmax layer, and we used back propagation to update the network weights for training. All layers of the network were fine-tuned using the same global learning rate of 0.001 and a decay factor of 16 every 30 epochs. We used RMSProp [27], which can speed up first-order gradient descent methods, with a decay of 0.9, momentum of 0.9, and epsilon of 0.1. We used Google's TensorFlow deep learning framework to train, validate, and test our network.

## 3. Results

### *3.1. Confusion Matrix*

Precision: The precision metric represents the correctly predicted labels out of the total true predictions. The precision achieved for every label is shown in Table 5.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

where TP and FP represent true positive and false positive.

**Table 5.** Precision of IDFNP for Every Taxonomy.

| **Taxonomy** | **N** | **L1** | **L2** | **L3** | **R1** | **R2** | **R3** |
|---|---|---|---|---|---|---|---|
| Precision | 0.974 | 0.956 | 0.980 | 0.960 | 0.969 | 0.959 | 0.951 |

Sensitivity: The sensitivity metric is used to quantify the cases that are predicted correctly (i.e., the number of predicted labels over all positive observations). IDFNP's sensitivity of every label is shown in Table 6.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{4}$$

where TP and FN represent true positive and false negatives, respectively.

**Table 6.** Sensitivity for Every Taxonomy.

| **Taxonomy** | **N** | **L1** | **L2** | **L3** | **R1** | **R2** | **R3** |
|---|---|---|---|---|---|---|---|
| Sensitivity | 0.974 | 0.977 | 0.937 | 0.993 | 0.961 | 0.933 | 0.978 |

Accuracy: This metric which represents correct predictions out of total predictions:

$$\text{Accuracy} = \frac{TP + FN}{TP + FN + FP + TN} = 97.5\% \tag{5}$$

where TP, TN, FP, and FN represent true positive, true negative, false positive and false negatives, respectively.

Figure 4 shows the confusion matrix of our method over the seven classes of predicted labels. Element of each confusion matrix represents the empirical probability of predicting class given that the ground truth

By analyzing the confusion matrix, one can observe that the proposed method can predict the FNP types well. The highest classification accuracy was 0.993, achieved for L3, while the lowest classification accuracy was 0.933 for R2. It can be seen that the accuracy is very high for the most serious disease conditions (R3 and L3), but the accuracy is not very high for intermediate disease conditions (R2 and L2). The overall accuracy was 97.5%.was class.

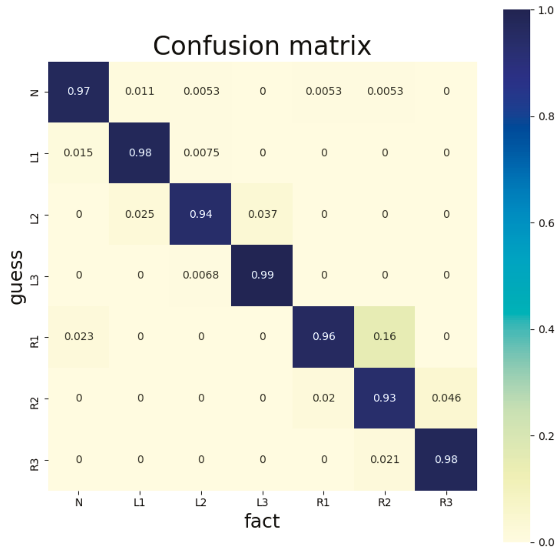


**Figure 4.** Confusion Matrix.

## *3.2. Comparison with Previous Methods and Neurologist Classification*

In this study, we divided all the movements (MV0, MV1, etc.) into different levels (N, L1, L2, et al.). In the process of specific training, we did not separate the different movements and did not test them accordingly. We believe that the FNP grading should not be performed by the movements. When FNP images are input into our system, the movement type does not need to be identified, as this is another deep learning topic; the output of our system is the FNP grading of the image. In our case, the accuracy for all movements was 97.5%.

To conclusively validate the algorithm, we used our previous method [25] for FNP quantitative assessment to compare validity with IDFNP. Meanwhile, neurologists classified the unlabeled FNP images. In this task, the IDFNP achieved 97.5% classification accuracy based on all movement, while our previous method for FNP quantitative assessment achieved 79.2–98.7% accuracy. Apart from MV0 RgAs, this method achieves a maximum of 94.4% in the other 13 ways of measuring FNP (Table 7).

**Table 7.** Comparison with previous method and neurological agreement.

| Movements | Previous Method Accuracy on RgAs | Previous Method Accuracy on AnAs | IDFNP CNN Accuracy | Neurological Agreement |
|---|---|---|---|---|
| MV0 | 98.7% | 80.6% | | 98.0% |
| MV1 | 94.4% | 81.9% | | 97.3% |
| MV2 | 93.1% | 80.6% | | 97.5% |
| MV3 | 94.4% | 79.2% | 97.5% | 97.4% |
| MV4 | 93.1% | 81.9% | | 97.1% |
| MV5 | 94.4% | 84.7% | | 97.7% |
| MV6 | 94.4% | 96.1% | | 98.0% |

We asked neurologists to diagnose each FNP image again when we went through the whole set; the double diagnosis agreement for the side affected by FNP reached 100%, while the double diagnosis agreement for the FNP degree ranged between 97.1% and 98.0%. Neurological agreement represents consistent neurological classification for FNP. As the images in the validation set were labeled by neurologists, but not necessarily confirmed by them, this metric is inconclusive, and instead actually shows that the CNN is learning relevant information.

### *3.3. Comparison with Other Computer-Aided Analysis Systems*

Sajid et al. [24] used a CNN model to classify face images with FNP into the five distinct degrees established by House and Brackmann. Sajid used GAN to prevent overfitting in training (Column 3, VGG-16 Net with GAN). Neely [28] used a computerized objective measurement of facial motion to obtain diagnosis of facial paralysis; using a standardized classification method, he achieved an accuracy of 95% (Columns 4). HC et al. [23] used optical-flow tracking and texture analysis methods to solve the problem. They used advanced image processing technology to capture the asymmetry of facial movements by analyzing the patients' video data and then used several different classification methods to diagnose FNP. The result is shown in Table 2 (Columns 5–6, RBF with 0/1 disagreement). Wang et al. [29,30] presented a novel method for grading facial paralysis integrating both static facial asymmetry and dynamic transformation factors. Wang used an SVM with the RBF kernel function to quantify the static facial asymmetry on images using five of the six facial movements (MV1–6), but they did not measure accuracy of MV0. The results are shown in column 7 of Table 8.

**Table 8.** Comparison with the other computer-aided analysis systems.

| Movements | IDFNP CNN Accuracy | VGG-16 Net with GAN | Neely's Method | RBF with 0 Disagreement | RBF with 1 Disagreement | SVM with the RBF |
|---|---|---|---|---|---|---|
| MV0 | | | | 44.36% | 92.26% | |
| MV1 | | | | 41.27% | 93.11% | 97.56% |
| MV2 | | | | 68.71% | 93.23% | 92.55% |
| MV3 | 97.5% | 92.60% | 95% | 49.80% | 94.18% | 91.36% |
| MV4 | | | | 49.80% | 94.18% | 95.40% |
| MV5 | | | | 61.78% | 86.31% | 93.36% |
| MV6 | | | | 49.80% | 94.18% | 95.40% |

### *3.4. Comparison with Other Deep Convolution Neural Networks Models*

Because our dataset's scale is not large enough to train models directly, for every model compared, we removed the final classification layer from the network, retrained it with our dataset, and leveraged the natural image features learned by the ImageNet pretrained network, a technique known as transfer learning. We chose Inception-v3, Inception-v4, Inception-ResNet-v1, Inception-ResNet-v2, DeepID, and ResNet, which in recent years have shown the best results in image classification. The results are shown in Table 9.

**Table 9.** Comparison of IDFNP with other classification CNNs.

| Network | Accuracy for FNP Dataset | Network | Accuracy for FNP Dataset |
|---|---|---|---|
| Inception-ResNet-v1 | 95.2% | Inception-ResNet-v2 | 95.7% |
| Inception-v3 | 93.3% | DeepID | 92.5% |
| Inception-v4 | 95.3% | ResNet | 95.0% |
| **IDFNP** | | **97.5%** | |

For accuracy, IDFNP CNN outperforms all the other CNNs for the FNP dataset. All the other CNNs were designed for the ImageNet Challenge Database, which has 1000 object classes and are optimized for image classification, which is quite relevant for the present application. Our original plan for diagnosing FNP was to use transfer learning with Inception-ResNet-v2 directly. However, the result did not match the accuracy of neurologists. Considering that FNP classification is a face classification, combining DeepID CNN with Inception-v3 CNN improves accuracy.

## 4. Discussion

As we see from Table 7, neurological agreement exceeds our method in MV2, MV5, and MV6. However, neurologists take too long examining FNP images, as each such examination takes at least

10 s. Our method takes a few milliseconds per FNP image and is thus more efficient, while its accuracy is comparable to that of neurologists. Our previous method takes much longer per FNP image by calculating facial asymmetry with traditional computational methods, while only its accuracy in MV0 on RgAs is higher. Furthermore, our previous method requires more standard images like face angle, image clarity, and lighting conditions.

As we see from Table 8, the accuracy of FNP classification when using Sajid's method was 92.6%. The accuracy of FNP in Neely's method [28] is 95%, which is lower than our method. HC [28] used RBF with 0/1 disagreement to measure accuracy of FNP movements. Even with 1 disagreement, which allows for more experimental errors, the result is significantly worse than ours. Wang [29,30] used SVM with RBF to measure accuracy. The result showed our method is better than their method in MV2–6. In MV1, their accuracy is not much higher than ours. Although they didn't calculate the accuracy of MV0, we can still see from the rest of the results that our method yields superior results.

As we see from Table 9, these models have strong generalization ability for different datasets, but because their design was optimized for their main, that is, image classification, the final training results of these models are not as good as our model. We also see that Inception-v3, upon which our own design was based, achieved only 93.3% accuracy. Therefore, there is still considerable potential for the optimization of this excellent image classification model for specific applications, especially with residual network derivatives like Inception-ResNet-v2.

Meanwhile, on the basis of our findings, clinicians can quickly obtain the degree of facial paralysis according to different facial movements. Clinicians can make a prediagnosis of facial nerve paralysis based on patients' facial movements, which will be used as a reference for their final diagnosis. For example, the result of one patient in MV1 (Eye closed), MV2 (Eyebrows raised), and MV4 (Grinning) was L3, and the result of the patient in other movements was N or L1, which corresponds to a prediagnosis that severe paralysis is present in the in left orbicularis oculi muscle.

## 5. Conclusions

In this paper, we presented a neural network model called IDFNP for FNP image classification, which uses a deep neural network and can achieve accuracy which is comparable to that of neurologists. Key to the performance of the model is an FNP annotated dataset and a deep convolutional network which can classify facial nerve paralysis and facial nerve paresis effectively and accurately. IDFNP combines Inception-v3, which achieves a great result in image classification, and DeepID, which is highly efficient in facial recognition.

The contributions of our method can be summarized as follows: Firstly, a symmetry-based annotation scheme for FNP images with seven different classes is presented. Secondly, using deep neural network on FNP images and cropping the face from the FNP images can eliminate facial deformation for FNP patients and minimize the influence of environmental factors. Thirdly, transfer learning avoids overfitting effectively for a limited range of FNP images. Combining an image classification CNN, such as Inception-v3, and a face recognition CNN like DeepID improves accuracy for the FNP dataset and achieves the same diagnostic accuracy as a neurologist. Fourthly, our method is validated against the performance of other well-known methods, which serves as proof that IDFNP is suitable for FNP classification and can effectively assist neurologists in clinical diagnosis.

In terms of clinical diagnosis, future work will be needed to apply IDFNP performance to other facial diseases or diseases which can be identified visually. On the one hand, more detailed diagnosis of facial paralysis would further aid neurologists in their work. In the future, we plan to undertake a more in-depth study of the position and the degree of disease. On the other hand, we can extend our findings to other conditions. For example, one of the symptoms of a stroke is facial asymmetry, which is very similar to the symptoms of FNP. If IDFNP can diagnose strokes and distinguish various degrees of facial stroke images and facial nerve paralysis images, then preventive treatment for strokes based on facial images can be realized. Given that modern smartphones and PCs are power tools of deep learning, with the help of the IDFNP results, citizens will have an enhanced ability to obtain an automated assessment for these diseases that may prompt them to visit a specialized physician.

The evaluation results produced by our methods are mostly consistent with the subjective assessment of doctors. Our methods can help clinicians to decide on a specific therapy for each patient, and for the most affected region of the face as reference.

Given that more and more FNP patients are being treated, high-accuracy diagnosis from FNP images can save expert clinicians and neurologists considerable time and decrease the frequency of misdiagnosis. Furthermore, we hope that this technology will enable greater widespread use of FNP images through photography as a diagnostic tool in places where access to a neurologist is limited.

**Author Contributions:** Conceptualization, A.S. and Z.W.; Data curation, A.S., Z.W., and X.D. (Xuehai Ding); Formal analysis, X.D. (Xuehai Ding); Funding acquisition, A.S.; Investigation, X.D. (Xuehai Ding); Methodology, A.S.; Project administration, A.S.; Resources, A.S. and X.D.; Software, Z.W. and Q.H.; Supervision, X.D. (Xuehai Ding); Validation, Z.W., Q.H., and X.D. (Xinyi Di); Visualization, Q.H.; Writing—original draft, Z.W.; Writing—review & editing, Z.W.

**Funding:** This research was funded by the National Natural Science Foundation of China (Grant No. 91630206).

**Acknowledgments:** This study is financially supported by the National Natural Science Foundation of China (Grant No. 91630206). Their support is greatly appreciated.

**Conflicts of Interest:** All authors declared that they have no conflict of interest.

## References

1. House, J.W.; Brackmann, D.E. Facial nerve grading system. *Laryngoscope* **2010**, *93*, 1056–1069.
2. Fields, M.J.; Peckitt, N.S. Facial nerve function index: A clinical measurement of facial nerve activity in patients with facial nerve palsies. *Oral Surg. Oral Med. Oral Pathol.* **1990**, *69*, 681–682. [CrossRef]
3. Ross, B.R.; Fradet, G.; Nedzelski, J.M. *Development of a Sensitive Clinical Facial Grading System*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 380–386.
4. Facs, J.G.N.M.; Cherian, N.G.; Dickerson, C.B.; Nedzelski, J.M. Sunnybrook facial grading system: Reliability and criteria for grading. *Laryngoscope* **2010**, *120*, 1038.
5. Vrabec, J.T.; Backous, D.D.; Djalilian, H.R.; Gidley, P.W.; Leonetti, J.P.; Marzo, S.J.; Morrison, D.; Ng, M.; Ramsey, M.J.; Schaitkin, B.M. Facial Nerve Grading System 2.0. *Otolaryngol. Head Neck Surg.* **2009**, *140*, 445–450.
6. Anguraj, K.; Padma, S. Analysis of Facial Paralysis Disease using Image Processing Technique. *Int. J. Comput. Appl.* **2013**, *54*, 1–4. [CrossRef]
7. Neely, J.G.; Joaquin, A.H.; Kohn, L.A.; Cheung, J.Y. Quantitative assessment of the variation within grades of facial paralysis. *Laryngoscope* **1996**, *106*, 438–442. [CrossRef]
8. Helling, T.D.; Neely, J.G. Validation of objective measures for facial paralysis. *Laryngoscope* **1997**, *107*, 1345–1349. [CrossRef]
9. Neely, J.G. Advancement in the evaluation of facial function. *Adv. Otolaryngol.* **2002**, *15*, 109–134.
10. Mcgrenary, S.; O'Reilly, B.F.; Soraghan, J.J. Objective grading of facial paralysis using artificial intelligence analysis of video data. In Proceedings of the IEEE Symposium on Computer-Based Medical Systems, Dublin, Ireland, 23–24 June 2005; pp. 587–592.
11. He, S.; Soraghan, J.; O'Reilly, B. Biomedical Image Sequence Analysis with Application to Automatic Quantitative Assessment of Facial Paralysis. *Eurasip J. Image Video Process.* **2007**, *2007*, 081282. [CrossRef]
12. Wachtman, G.S.; Liu, Y.; Zhao, T.; Cohn, J.; Schmidt, K. Measurement of Asymmetry in Persons with Facial Paralysis. Available online: https://www.ri.cmu.edu/publications/measurement-of-asymmetry-in-persons-with-facial-paralysis/ (accessed on 15 November 2018).
13. Lee, H.Y.; Park, M.S.; Byun, J.Y.; Ji, H.C.; Na, S.Y.; Yeo, S.G. Agreement between the Facial Nerve Grading System 2.0 and the House-Brackmann Grading System in Patients with Bell Palsy. *Clin. Exp. Otorhinolaryngol.* **2013**, *6*, 135–139. [CrossRef]
14. Yann, L.C.; Yoshua, B.; Geoffrey, H. Deep learning. *Nature* **2015**, *521*, 436–444.
15. Esteva, A.; Kuprel, B.; Novoa, R.A.; Ko, J.; Swetter, S.M.; Blau, H.M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **2017**, *542*, 115–118. [CrossRef]

16. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
17. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
18. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
19. 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, 26 June–1 July 2016; pp. 770–778.
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef]
21. Sun, Y.; Wang, X.; Tang, X. Deep Learning Face Representation from Predicting 10,000 Classes. In Proceedings of the Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1891–1898.
22. Rajpurkar, P.; Hannun, A.Y.; Haghpanahi, M.; Bourn, C.; Ng, A.Y. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *arXiv*, 2017; arXiv:1707.01836.
23. Hoochang, S.; Roth, H.R.; Gao, M.; Lu, L.; Xu, Z.; Nogues, I.; Yao, J.; Mollura, D.; Summers, R.M. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Trans. Med Imaging* **2016**, *35*, 1285.
24. Sajid, M.S.T.; Baig, M.; Riaz, I.; Amin, S.; Manzoor, S. Automatic Grading of Palsy Using Asymmetrical Facial Features: A Study Complemented by New Solutions. *Symmetry* **2018**, *10*, 242. [CrossRef]
25. Song, A.; Xu, G.; Ding, X.; Song, J.; Xu, G.; Zhang, W. Assessment for facial nerve paralysis based on facial asymmetry. *Australas. Phys. Eng. Sci. Med.* **2017**, *40*, 1–10.
26. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
27. Tieleman, T.; Hinton, G. *Lecture 6.5-RMSProp, COURSERA: Neural Networks for Machine Learning*; University of Toronto: Toronto, ON, USA, 2012.
28. Neely, J.G.; Wang, K.X.; Shapland, C.A.; Sehizadeh, A.; Wang, A. Computerized Objective Measurement of Facial Motion: Normal Variation and Test-Retest Reliability. *Otol. Neurotol.* **2010**, *31*, 1488–1492. [CrossRef]
29. Wang, T.; Zhang, S.; Dong, J.; Liu, L.A.; Yu, H. Automatic evaluation of the degree of facial nerve paralysis. *Multimed. Tools Appl.* **2016**, *75*, 11893–11908. [CrossRef]
30. Wang, T.; Dong, J.; Sun, X.; Zhang, S.; Wang, S. Automatic recognition of facial movement for paralyzed face. *Biomed. Mater. Eng.* **2014**, *24*, 2751–2760.

*Essay*

# Chinese Text Classification Model Based on Deep Learning

**Yue Li *, Xutao Wang and Pengjian Xu**

School of Computer Science and Technology, Donghua University, Shanghai 201620, China; 2161704@mail.dhu.edu.cn (X.W.); 2171774@mail.dhu.edu.cn (P.X.)
* Correspondence: frankyueli@dhu.edu.cn

Received: 18 October 2018; Accepted: 12 November 2018; Published: 20 November 2018

**Abstract:** Text classification is of importance in natural language processing, as the massive text information containing huge amounts of value needs to be classified into different categories for further use. In order to better classify text, our paper tries to build a deep learning model which achieves better classification results in Chinese text than those of other researchers' models. After comparing different methods, long short-term memory (LSTM) and convolutional neural network (CNN) methods were selected as deep learning methods to classify Chinese text. LSTM is a special kind of recurrent neural network (RNN), which is capable of processing serialized information through its recurrent structure. By contrast, CNN has shown its ability to extract features from visual imagery. Therefore, two layers of LSTM and one layer of CNN were integrated to our new model: the BLSTM-C model (BLSTM stands for bi-directional long short-term memory while C stands for CNN.) LSTM was responsible for obtaining a sequence output based on past and future contexts, which was then input to the convolutional layer for extracting features. In our experiments, the proposed BLSTM-C model was evaluated in several ways. In the results, the model exhibited remarkable performance in text classification, especially in Chinese texts.

**Keywords:** Chinese text classification; long short-term memory; convolutional neural network

## 1. Introduction

Motivated by the development of Internet technology and the progress of mobile social networking platforms, the amount of textual information is growing rapidly on the Internet. Given the powerful real-time nature of Internet platforms, great potential value is hidden in such textual information; however, effective organization and management is in high demand presently. Text classification, known as an effective method for text information organization and management, is widely employed in the fields of information sorting [1], personalized news recommendation, sentiment analysis [2], spam filtering, user intention analysis, etc.

Machine-learning-based methods, including naive Bayes, support vector machine, and k-nearest neighbors, are generally adopted by traditional text classification. However, their performance depends mainly on the quality of hand-crafted features. Compared with the methods of machine learning, the method of deep learning proposed in 2006 is deemed as an effective method for feature extraction. Moreover, an increasing number of scholars have applied commonly used neural networks, including the convolutional neural network (CNN) and the recurrent neural network (RNN), to text classification.

Among the two, RNN has attained remarkable achievement in handling serialization tasks. As RNN is equipped with recurrent network structure which can be used to maintain information, it can better integrate information in certain contexts. For the purpose of avoiding the problem of gradient exploding or vanishing in a standard RNN, long short-term memory (LSTM) [3] and other variants [4] have been designed for the improvement of remembering and memory accesses. Living up to expectations, LSTM does show a remarkable ability in the processing of natural language. Moreover, the other popular neural network, CNN, has also displayed a remarkable performance in computer vision [5], speech recognition, and natural language processing [6] because of its remarkable capability in capturing local correlations of spatial or temporal structures. In terms of natural language processing, CNN is able to extract n-gram features from different positions of a sentence through convolutional filters and then it learns both short- and long-range relations through the operations of pooling.

LSTM [3] is good at dealing with serialization tasks but poor in the ability to extract features, performs well in extracting features but lacks the ability to learn sequential correlations. Therefore, in the paper, both the CNN and the specific RNN architecture—bidirectional long short-term memory (BLSTM)—are combined together to establish a new model named as the BLSTM-C model for text classification. The BLSTM is employed firstly to capture the long-term sentence dependencies and then CNN is adopted to extract features for sequence modeling tasks. In addition, our model is evaluated by applying it to Chinese text classification. The model is applied to both the English and Chinese language and then corresponding effects are compared with each other. It turns out that our model is more suitable for the Chinese language. Furthermore, it is also shown through our evaluation that our BLSTM-C model achieves remarkable results and it also outperforms a wide range of baseline models.

## 2. Related Work

It is widely acknowledged that deep-learning-based neural network models have achieved great success in natural language processing. This paper focuses on establishing a new model that is able to obtain better results in the classification of Chinese text. To ensure that the computer can understand human language, the first step of text classification usually goes to representing the text with vectors which will later feed into the neural network. Therefore, the quality of the representation is doomed to play a quite significant role in the classification. For the final aim of obtaining a better representation of text, TF-IDF (Term Frequency–Inverse Document Frequency) and bag-of-words were employed in early research. Bag-of-words treats texts as unordered sets of words and each word of them is represented by a one-hot vector, a sparse vector in the size of the vocabulary, with 1 in the entry representing the word and 0 in all other entries [1]. Accordingly, this vector loses both the word order and syntactic feature. Mikolov came up with the idea of distributed representations of words and paragraphs, and it is shown by relevant experiments that word and paragraph vectors outperform bag-of-words models as well as other techniques for text representations [7,8].

In many works on text representation learning published recently, it is known that there are generally two popular neural network models that had achieved their remarkable performance—CNN and RNN.

The ability of CNNs [5] in extracting features from inputs, such as images, is outstanding and it has also achieved remarkable result in image classification. In such process, 2D convolution and 2D pooling operation are usually used to represent image input [5,9]. As for text input, Kalchbrenner utilized 1D convolution to perform feature mapping and then applied 1D max pooling operation over the time-step dimension to obtain a fixed-length output [6,10]. Moreover, in 2017, Conneau et al. adopted a very deep CNN in the tasks of text classification by pushing the depth to 29 convolutional layers [11].

RNN, as what is indicated by its name, is known as a kind of neural network that is equipped with a recurrent structure, for which RNN is capable of preserving sequence information over time. In addition, this feature enables that RNNs is applicable for serialization tasks such as text classification and speech recognition. Furthermore, Tai et al. [12] proposed a tree-LSTM, a variant of RNN allowing for richer network topologies where each LSTM unit is able to incorporate information collected from multiple child units. In addition, Zhou et al. [13] achieved success in extracting meaningful features from documents automatically by combining bi-directional LSTM with an attention mechanism.

Our work focuses mainly on Chinese text classification, which is known as a completely different language from English. In English, words are generally separated by spaces and an independent meaning is available for each word, while Chinese words, on the contrary, have no spaces to separate them. In order to solve this problem, Zhang et al. [14] put forward an HHMM-based Chinese lexical analyzer, ICTCLAS, which actually showed the effectiveness of class-based segmentation HMM(Hidden Markov Model). Furthermore, Ma et al. employed a deep learning method—LSTM—to conduct Chinese word segmentation and achieved better accuracy in many popular datasets in comparison with the models based on more complex neural network architectures. As what is stated above, LSTM performs poorly in extracting features, while CNN lacks the ability to learn sequential correlations. In this paper, a new model integrating BLSTM with CNN is established for Chinese news classification. There has already been research that focuses on computer vision tasks such as image captions by combining CNN with LSTM. Moreover, Zhou et al. [15] also proposed a C-LSTM model that integrates CNN with an LSTM network for sentence modeling. Most of these models firstly apply CNN to data inputting to extract features and then feed them to the LSTM layer. However, the approach adopted by us is totally different: bi-directional LSTM is employed capture long-term sentence dependencies and then CNN is applied to extracting features for classification. Our experiment on the classification of Chinese news shows that our model outperforms the other related sequence-based models.

## 3. Hybrid Deep-Learning Model

The whole process of classification is shown in Figure 1. This chapter will describe each layer in the subsections and combine them as the BLSTM-C model at the end of this chapter.

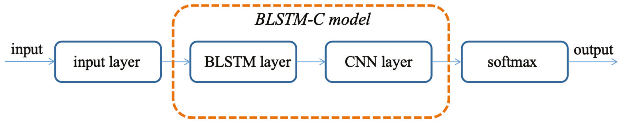


**Figure 1.** Classification process.

### 3.1. Input Layer

The first layer is the input layer that does some processing on the input. This layer can be divided into two parts: remove the stop words and segmentation. The first step removes the stop words from the article so that the information in the text will be more concentrated. The second step is a unique processing for Chinese language. Since Chinese words do not have a space to separate them, it would be necessary to segment them into words manually so that we can represent each word as vector.

#### 3.1.1. Remove The Stop Words

In computing, stop words are words which are filtered out before or after processing of natural language data (text). Any group of words can be chosen as the stop words for a given purpose. For some search engines, these are some of the most common, short function words, such as 'the', 'is', 'at', 'which', and so on.

#### 3.1.2. Segmentation

Chinese language is a unique language that is completely different from English. In English, word and word are separated by space and each word stands for independent meaning. On the contrary, Chinese words do not have a space to separate them. Furthermore, although each word has its independent meaning, their meaning is changed when the words are put together. Therefore, it is important and difficult to separate the words based on the context. Wrong Segmentation will totally change the sentence's meaning and increase the difficulty of classification.

After comparing the most commonly used tools for Chinese word segmentation, we finally choose "Jieba", which is built to be the best Python Chinese word segmentation module. The mainly algorithms for it are as follows:

- Achieve efficient word graph scanning based on a prefix dictionary structure [16]. Build a directed acyclic graph (DAG) for all possible word combinations.
- Use dynamic programming to find the most probable combination based on the word frequency.
- For unknown words, an HMM-based model [17] is used with the Viterbi algorithm.

For the Chinese words, it uses four states (BEMS) to distinguish them: B(begin), E(end), M(middle), and S(single). In addition, after training on large quantities of corpora, it gets three probability tables: TransProbMatrix, Emission Probability Matrix, and Initial State Matrix. Then, for a sentence that needs to be segmented, the HMM model uses a Viterbi algorithm to obtain the best 'BEMS' sequence that begins with a 'B'-word and ends with an 'E'-word.

Assume that, given the HMM state space S, there are k states, the probability of the initial state $i$ is $\pi_i$, and the transition probability from the state $i$ to the state $j$ is $a_{ij}$. Let the observed output be $y_1$, ..., $y_T$. The most likely state sequence $x_1$, ..., $x_T$ that produced the observation is given by the recurrence relation:

$$V_{i,j} = P(y_l|k) \cdot \pi_k, \tag{1}$$

$$V_{t,k} = max_{x \in S}(P(y_t|k) \cdot a_{x,k} \cdot V_{t-1,x}). \tag{2}$$

Here, $V_{t,k}$ is the probability of the most likely state sequence to correspond to the first $t$ observations with a final state of $k$. The Viterbi path can be obtained by saving the backward pointer to remember the state x used in the second equation and declaring a function $Prt(k,t)$ that returns the $x$ value to be used if $t > 1$ or returns a $k$ value if $t = 1$:

$$x_T = argmax_{x \in S}(V_{T,x}), \tag{3}$$

$$x_{t-1} = Prt(x_t, t). \tag{4}$$

### 3.2. BLSTM Layer

Long Short-term Memory (LSTM) [3] is developed on the basis of Recurrent Neural Network (RNN) to solve the problems related to gradient vanishing or exploding. The mainly idea used by it is adding "gate" to the Recurrent Neural Network for the ultimate purpose of controlling the passing data. Generally speaking, a common architecture of LSTM units is composed of a memory cell, an input gate, an output gate and a forget gate. LSTM is shown in the form of a chain that is constructed by repeating modules of neural networks. With information stored inside, the memory cell runs across the whole chain. In addition, the other three gates are mainly designed to control whether to add or block the information to the memory cell.

With the output from $h_{t-1}$, the old hidden state, and the input from $x_t$, the current moment, the gates determine how to update the current memory cell and $h_t$, the current hidden state. The output of the forget gate represents the proportion of information that will be kept, while the input gate is mainly responsible for the addition of information to the memory cell. Moreover, this addition operation is made up of three parts. Firstly, a sigmoid layer is used to regulate the information that is required to be added to the memory cell. Secondly, the *tanh* function is adopted to obtain a vector through $h_{t-1}$ and $x_t$. Finally, multiply these two values obtained and feed them to the memory cell. The output gate is responsible for the task of selecting useful information from the memory cell to output. For this purpose, it should firstly create a vector by applying the *tanh* function to the cell state and then regulate the information from $h_{t-1}$ to $x_t$ and multiply it by the vector created before. In this way, the output for this moment is obtained.

The LSTM transition functions are defined as follows:

$$i_t = \sigma(W_t\,[h_{t-1}, x_t] + b_i), \tag{5}$$

$$f_t = \sigma(W_f\,[h_{t-1}, x_t] + b_f), \tag{6}$$

$$o_t = \sigma(W_o\,[h\cdot, x\cdot] + b_o), \tag{7}$$

$$\tilde{c_t} = tanh(W_c[h_{t-1}, x_t] + b_f), \tag{8}$$

$$c_t = f_t \bullet c_{t-1} + i_t \bullet \tilde{c_t}, \tag{9}$$

$$h_t = o_t \bullet tanh(c_t). \tag{10}$$

$\sigma$ refers to the logistic sigmoid function that has an output in [0, 1], *tanh* indicates the hyperbolic tangent function that has an output in [−1, 1], and $\bullet$ denotes the elementwise multiplication. At the current time $t$, $h_t$ refers to the hidden state, $f_t$ represents the forget gate, $i_t$ indicates the input gate, and $o_t$ denotes the output gate. $W_t$, $W_o$, and $W_f$ represent the weight of these three gates, respectively, while $b_t$, $b_o$, $b_f$ refers to the biases of the gates.

As for BLSTM, it is regarded as an extension of the undirectional LSTM, and it not only adds another hidden layer but also connects with the first hidden layer in the opposite temporal order. Because of its structure, BLSTM can process the information from both the past and the future. Therefore, BLSTM is adopted to capture the information of the text input in this paper.

## 3.3. Convolutional Neural Networks Layer

There are only two dimensions in the input, among which $x_i \in \Re^d$ represents the d-dimensional vector for the $i$-th word in the sentence, while $x_i \in \Re^d$ denotes the input sentence. Moreover, L refers to the length of the sentence. Furthermore, one-dimensional convolution is employed to extract features from the output of LSTM layer.

A filter sliding over the input sequence is adopted by the one-dimensional convolution to detect features from different positions. Vector of the word in the sliding filter is denoted by $x_j$, $x_{j+1}$, $\cdots$, $x_{j+k-1}$, respectively. The window vector can be represented by the formula as follows:

$$W_j = [x_j, x_{j+1}, \cdots, x_{j+k-1}]. \tag{11}$$

The window vectors that are related to with the word $x_j$ are: $w_{j-k+1}$, $w_{j-k+2}$, $\cdots$, and $w_j$, respectively. For each $w_j$, window vector, its feature map can be expressed by the formula follows:

$$c_j = f(w_j \circ m + b), \tag{12}$$

where $\circ$ refers to the dot product, $b \in R$ represents a bias term and $f$ denotes a nonlinear transformation function that can be sigmoid, and hyperbolic tangent, etc. In our experiment, ReLU is chosen as the nonlinear function. In our model, n filters are adopted by us to produce the feature maps as follows:

$$\mathbf{W} = [c_1, c_2, \cdots, c_n]. \tag{13}$$

In the formula above, $c_i$ refers to the feature map generated by the $i$-th filter. The convolution layer may have multiple filters of the same size to learn complementary features, or multiple kinds of filters with different sizes.

Then, a max-over pooling is applied to this feature map for the purpose of obtaining a vector of fixed length for classification. As for the pooling operation, it is adopted to extract maximum value from the matrix (feature map). After each convolution, a max-pool layer is added to extract the most significant elements in each convolution and then they are turned into feature vectors. It is common to periodically insert a pooling layer in-between successive Conv layers in a ConvNet architecture, which can progressively reduce the spatial size of the representation so as to reduce the amount of parameters and computation in the network, thus controlling the overfitting. Two common pooling operations, max pooling and average pooling, are shown in Figure 2. Max pooling chooses the max value in the filter as the new value in the new matrix while average pooling adopts the average value of all the numbers existing in the filter.

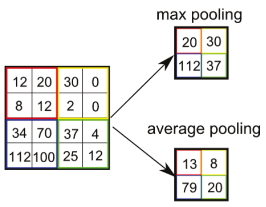


**Figure 2.** Pooling operation.

## *3.4. Proposed BLSTM-C Model*

As shown in Figure 3, our model begins with a BLSTM layer to obtain a sequence output on the basis of the past context and the future context. Then, this sequence is fed to the CNN layer that is utilized to extract features from the previous sequence. After that, a max-over pooling layer is adopted to obtain a fixed length vector that is fed to the output layer that employs softmax function to classify the input. Blocks of the same color in the feature map layer and the window feature sequence layer correspond to features of the same window.

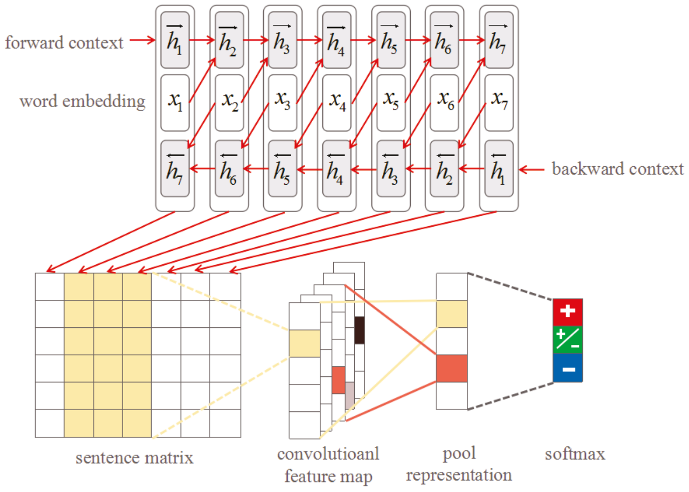


**Figure 3.** The architecture of the BLSTM-C model.

In the theory of probability, the output of the softmax function can be employed to represent a categorical distribution, that is, a probability distribution over K (number of different possible outcomes). As for our experiment, the probability distribution obtained will be over the categories of dataset. Moreover, the biggest one is the category that this input text belongs to. The function is shown as follows:

$$\sigma : \mathbb{R}^K \rightarrow \left\{ z \in \mathbb{R}^K \mid z_i > 0, \sum_{i=1}^{K} = 1 \right\}, \tag{14}$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \tag{15}$$

## 4. Experiment

### *4.1. Datasets*

Our BLSTM-C model is evaluated on three datasets, and the summary statistics of the datasets obtained are shown as follows:

SST-1: Stanford Sentiment Treebank benchmark from Socher et al. [2]. This dataset is made up of 11,855 movie reviews and the reviews are split into train (8544), dev (1101) and test (2210), which aims to classify a review with fine-grained labels (very negative, negative, neutral, positive, and very positive).

SST-2: The same as SST-1, but the neutral reviews in it are removed and the binary labels (positive, negative) are adopted.

THUCNews: THUCNews is generated according to the historical data obtained from the subscription channel of Sina News RSS from 2005 to 2011. Based on the original classification system of Sina news, it is reintegrated and divided.

BBC: The English news dataset is originated from BBC news and it is adopted as the benchmarks for the research on machine learning. It is composed of 2225 documents from the BBC news website corresponding to stories from five topical areas from 2004 to 2005, including business, entertainment, politics, sport, and tech.

In all, eight categories of articles are selected by the main Chinese classification experiment for classification and they include politics, economy, stock market, technology, sports, education, fashion, and games, respectively.

For the comparison experiment on Chinese and English, five categories of articles are selected and they include business, entertainment, politics, sport, and tech.

### *4.2. Word Vector Initialization and Padding*

Firstly, word2vec is employed to pre-train on large unannotated corpora. Through this way, better generalization can be achieved on the basis of limited amount of training data. In addition, maxlen is also set up to denote the maximum length of the sentence. Then, for every sentence, the stopwords are removed and the top maxlen words occurring in the word2vec model are transformed into vectors. For those sentences which are shorter than maxlen, they are padded with '0' vectors by us. In this way, the fixed-length input can be obtained for our model.

### *4.3. Hyper-Parameter Setting*

For each dataset, 60% of the articles are randomly selected for training, 10% for validation and 30% for test. Moreover, the hyper-parameters are set up as follows.

The dimension of word vector is 300 for English word since the pre-trained word2vec model is chosen from Google for English, while the dimension of word vector for Chinese word is 250 because better representation on this configuration is obtained when the Chinese word2vec model is being trained by ourselves.

The maxlen of the sentence for SST-1 and SST-2 is 18 while the same parameter for THUCNews is 100. This maxlen parameter is established on the basis of the average length of the articles in each dataset. After lots of experiments, one BLSTM layer and one Convolutional layer are adopted by us when building our BLSTM-C model for all tasks. For the BLSTM layer, 50 hidden units are employed and the dropout rate obtained is 0.5. For Convolution layer, 64 convolutional filters with the window size of 5, and 1D pooling with the size 4 are employed.

## **5. Results**

As shown in Table 1, our model is compared with 14 well-performed models from different tasks. One of the tasks is sentiment classification (SST-1, SST-2) while the other one is category classification (THUCNews).

### *5.1. Overall Performance*

Both SST-1 and SST-2 datasets are employed to compare the performance of different methods. As shown in Table 1, our model is compared with some well-performed models from different areas, such as Support Vector Machine(SVM), Recursive Neural Network, Convolutional Neural Network, and Recurrent Neural Network. Specifically, for the Recursive Neural Network, what are chosen include MV-RNN: Semantic Compositionality through Recursive Matrix-Vector Spaces [18], RNTN: Recursive Deep Models for semantic compositionality over a sentiment treebank [2], and DRNN: Deep Recursive Neural Networks for compositionality in language [19]. For CNN, what are chosen include DCNN: a CNN for modeling sentences [6], CNN-nonstatic and CNN multichannel: Convolutional Neural Networks for sentence classification [10], and Molding-CNN: Molding CNNs for text, including nonlinear and non-consecutive convolutions [20]. For Recurrent Neural Networks, what are chosen include RCNN: Recurrent Convolutional Neural Networks for Text Classification [20], S-LSTM: Long Short-term Memory over recursive structures [21], BLSTM and Tree-LSTM: improved semantic representations from tree-structured Long Short-term Memory networks [12]. For other baseline methods, we use a Support Vector Machine, n-gram bag of words and a Paragraph Vector. In addition, LSTM and B-LSTM were also implemented by us for further comparison of category classification on our Chinese news dataset.

**Table 1.** Comparison with baseline models on Stanford Sentiment Treebank and THUCNews.

| Model | SST-1(%) | SST-1(%) | THUCNews(%) | Reported in |
|---|---|---|---|---|
| SVM | 40.7 | 79.4 | 77.5 | Socher et al.(2013b) [2] |
| NBoW | 42.4 | 80.5 | 75.5 | Le and Mikolov (2014) [8] |
| Paragraph Vector | 48.7 | 48.7 | 48.7 | Le and Mikolov (2014) [8] |
| DCNN | 48.5 | 86.8 | - | Kalchbrenner et al. (2014) [6] |
| CNN-non-static | 48.0 | 87.2 | - | Kim (2014) [10] |
| Modeling-CNN | **51.2** | 88.6 | - | Lei et al. (2015) [20] |
| CNN-multichannel | 47.4 | 88.1 | - | Kim (2014) [10] |
| RCNN | 47.21 | - | - | Lai et al. (2015) [20] |
| S-LSTM | - | 81.9 | - | Zhu et al. (2015) [21] |
| BLSTM | 49.1 | 87.5 | - | Tai et al. (2015) [12] |
| Tree-LSTM | 51.0 | 87.5 | - | Tai et al. (2015) [12] |
| MV-RNN | 44.4 | 82.9 | - | Socher et al. (2012) [18] |
| RNTN | 45.7 | 85.4 | - | Socher et al. (2013b) [2] |
| DRNN | 49.8 | 86.6 | - | Irsoy and Cardie (2014) [19] |
| LSTM | 47.1 | 87.0 | 83.4 | Our implementation |
| B-LSTM | 47.3 | 88.1 | 86.5 | Our implementation |
| CNN | 46.5 | 85.5 | 82.5 | Our implementation |
| BLSTM-C | 50.2 | **89.5** | **90.8** | Our implementation |

The table above shows that our BLSTM-C model achieves remarkable performance in two out of three tasks (numbers in bold represent the best results). In sentiment classification, our BLSTM-C model gets the best result in the SST-2 dataset while molding-CNN achieves the best performance in the SST-1 dataset. Although our model fails to beat the state-of-art ones, it still obtains an acceptable result which means that the model is feasible for various scenarios.

As for the classification of text category, our model outperforms other well-performed models, achieving outstanding results. Through the comparison between our model and single-layer LSTM, B-LSTM and LSTM models, it is found that our model does combine the advantages of both LSTM and CNN. Apart from successfully learning long-term dependencies, it extracts features from text, leading to better results. Although almost no human-designed features are employed in our model, it beats the state-of-the-art SVM that highly requires engineered features.

### *5.2. Comparison between English and Chinese*

To validate the ability of our model on different languages, our BLSTM-C model is compared with a simple LSTM model on the English news dataset as well as the Chinese news dataset that has the same categories as the English one. The five categories include technique, sports, business, entertainment and politics. For an English experiment, a Google pre-trained word2vec model is chosen to represent the English words with vectors. The English news dataset is originated from BBC news and it mainly serves as the benchmarks for machine learning research. It is composed of 2225 documents from the BBC news website, corresponding to stories from five topical areas from 2004 to 2005.

Simple LSTM is also adopted by us to compare the improvement of our BLSTM-C model on Chinese dataset with that of English dataset. Tables 2 and 3 show the results obtained from English while Tables 4 and 5 present the results gained from Chinese. The number displayed in the table represents the number of articles that are classified as within this category.

**Table 2.** Output of the simple LSTM on the English dataset.

| | **Tech** | **Sports** | **Politics** | **Entertainment** | **Business** |
|---|---|---|---|---|---|
| Tech | 145 | 2 | 1 | 46 | 13 |
| Sports | 0 | 264 | 0 | 0 | 0 |
| Politics | 1 | 6 | 178 | 2 | 7 |
| Entertainment | 2 | 0 | 0 | 180 | 0 |
| Business | 2 | 1 | 6 | 3 | 254 |
| Accuracy | | | | | 91.73% |

**Table 3.** Output of the BLSTM-C on the English dataset.

| | **Tech** | **Sports** | **Politics** | **Entertainment** | **Business** |
|---|---|---|---|---|---|
| Tech | 187 | 0 | 0 | 9 | 11 |
| Sports | 1 | 261 | 0 | 1 | 1 |
| Politics | 2 | 1 | 179 | 0 | 12 |
| Entertainment | 4 | 0 | 2 | 175 | 1 |
| Business | 5 | 0 | 7 | 0 | 254 |
| Accuracy | | | | | 94.88% |

**Table 4.** Output of the simple LSTM on the Chinese dataset.

| | **Tech** | **Sports** | **Politics** | **Entertainment** | **Business** |
|---|---|---|---|---|---|
| Tech | 163 | 3 | 4 | 27 | 10 |
| Sports | 3 | 250 | 7 | 1 | 3 |
| Politics | 3 | 2 | 178 | 1 | 10 |
| Entertainment | 1 | 7 | 0 | 173 | 1 |
| Business | 6 | 2 | 6 | 2 | 250 |
| Accuracy | | | | | 91.11% |

**Table 5.** Output of the BLSTM-C on the Chinese dataset.

| | **Tech** | **Sports** | **Politics** | **Entertainment** | **Business** |
|---|---|---|---|---|---|
| Tech | 200 | 0 | 0 | 5 | 2 |
| Sports | 1 | 258 | 0 | 5 | 0 |
| Politics | 2 | 1 | 184 | 1 | 6 |
| Entertainment | 3 | 5 | 0 | 172 | 2 |
| Business | 3 | 1 | 4 | 1 | 257 |
| Accuracy | | | | | 96.23% |

The tables show that our BLSTM-C model achieves better performance in both experiments, which means that our model is suitable for both Chinese and English languages. It is worth mentioning that our model gets a more significant improvement in the Chinese dataset, a 5.1% higher accuracy than that of a simple LSTM model, while the improvement on the English experiment is only 3.15%.

It can be concluded from the experiments that our BLSTM-C is more suitable for the Chinese language because of the unique structure of Chinese.

## *5.3. Performance Analysis*

Here, the impacts of different parameters on our model performance are analyzed.

- The length of the *maxlen*

  In the initialization and padding of word vectors, the parameter, maxlen, was set up to determine the length of the words that are chosen to represent the article. As for SST-1 and SST-2 datasets, the average length of articles are 18, and the length is so short that it is difficult to obtain the different influences of the article length. Therefore, the THUCNews dataset is selected for an experiment to find out the effect of article length. Figure 4 shows that different lengths of the articles result in different performance.

  The best result, 93.77%, occurs when 80 words are employed to represent the article, which is also deemed as the closest length to the average article length of the dataset. Once the maxlen is far greater than the average length of the articles, then the accuracy will decrease greatly because many more zero vectors will exist in the vectors of the article.

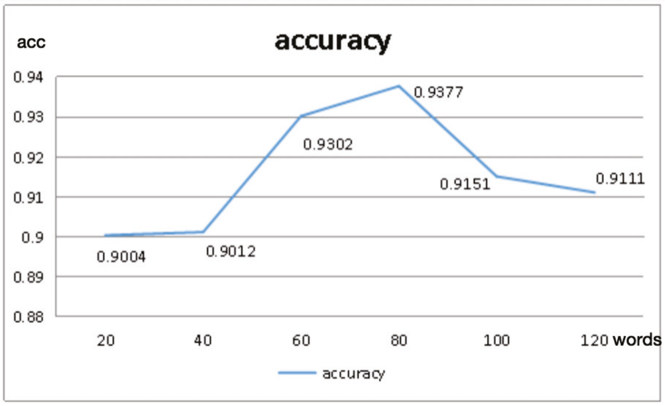


**Figure 4.** Accuracy vs. article length.

- The size of Convolutional Filter

  In Figure 5, different convolution filter configurations are adopted to present the prediction accuracy on the question classification. As for the horizontal axis, the number indicates convolutional kernel size, and bar charts of five different colors on each filter size represent different dimensions of the convolution output. For example, “S8” means that the size of the kernel is 8 while “F128” denotes that the dimension of the convolution output is 128. It is quite obvious that the dimension of 64 outperforms the other dimensions and the best result, the accuracy of 93.55%, occurs when the window size is 7.

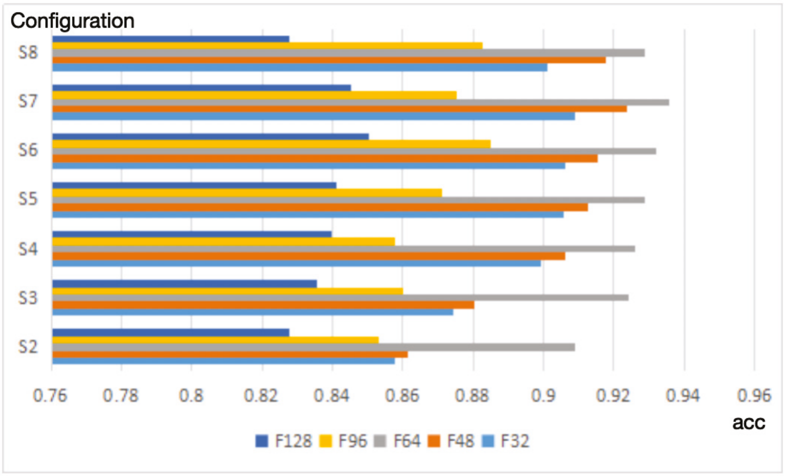


**Figure 5.** Accuracy vs. different filter configuration.

## 6. Conclusions

This paper mainly introduces a combined model called BLSTM-C that is made up of a bi-directional LSTM layer and a convolutional layer. The paper also shows its ability in information learning from both previous context and future context, as well as its competence to extract features. It is shown by the experiment results that this model performs remarkably on the tasks related to Chinese News classification, and it also outperforms CNN, RNN and other models on sentiment classification. Moreover, by running the experiments on similar dataset in Chinese and English, it is found out that our BLSTM-C model may have better performance in the Chinese language because the improvement shown in the Chinese language is more significant. To obtain better performance in Chinese news classification, the suitable parameters for this model are also explored and it is found that it is helpful to improve the results by setting the maxlen closest to the average article length and adopting a suitable window to detect the features.

Furthermore, the following suggestions may lead to better performance in future work. Firstly, it will be an interesting idea to deepen the neural network layer. Research on text classification by employing 29 convolutional layers gets satisfying results. Moreover, it is also common for Long Short-term Memory (LSTM) to be multi-layer. Secondly, the article will be more reasonable if every location, number, name and some other meaningless words are replaced with placeholders like '[location]', '[number]','[name]'. This operation will enable the article to be clearer for computer systems.

Therefore, in future work, these modifications will be tried in our experiment to see if it will achieve better performance.

**Funding:** This research was funded by the Natural Science Foundation of Shanghai Grant No. 16ZR1401100, and the National Key R&D Program of China, Grant No. 2017YFB0309800.

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

## References

1. Wang, S.; Manning, C.D. Aselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 8–14 July 2012; Volume 2, pp. 90–94.
2. Socher, R.; Perelygin, A.; Wu, J.Y.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013.

3. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
4. Cho, K.; Merrienboer, B.V.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
5. Krizhevsky, A.I.S.A.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
6. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. *arXiv* **2014**, arXiv:1404.2188.
7. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; Volume 2, pp. 3111–3119.
8. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), Beijing, China, 21–26 June 2014; pp. 1188–1196.
9. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
10. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
11. Conneau, A.; Schwenk, H.; Barrault, L.; Lecun, Y. Very deep convolutional networks for text classification. *arXiv* **2017**, arXiv:1606.01781.
12. Tai, K.S.; Socher, R.; Manning, C.D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv* **2015**, arXiv:1503.00075.
13. Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; Xu, B. Attention-based bidirectional long short-term memory networks for relation classification. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; p. 207.
14. Zhang, H.-P.; Yu, H.-K.; Xiong, D.-Y.; Liu, Q. Hhmm-based chinese lexical analyzer ictclas. In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing, SIGHAN '03, Sapporo, Japan, 11–12 July 2003; pp. 184–187.
15. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. A c-lstm neural network for text classification. *arXiv* **2015**, arXiv:1511.08630.
16. Zhu, Z.; Yin, H.; Chai, Y.; Li, Y.; Qi, G. A novel multi-modality image fusion method based on image decomposition and sparse representation. *Inf. Sci.* **2018**, *432*, 516–529. [CrossRef]
17. Lee, K.-F.; Hon, H.-W. Speaker-independent phone recognition using hidden Markov models. *IEEE Trans. Acoust. Speech Signal Process.* **1989**, *37*, 1641–1648. [CrossRef]
18. Socher, R.; Huval, B.; Manning, C.D.; Ng, A.Y. Semantic compositionality through recursive matrixvector spaces. In Proceedings of the Empirical Methods on Natural Language Processing, Jeju Island, Korea, 12–14 July 2012; pp. 1201–1211.
19. Irsoy, O.; Cardie, C. Deep recursive neural networks for compositionality in language. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2096–2104.
20. Lei, T.; Barzilay, R.; Jaakkola, T. Molding cnns for text: Non-linear, nonconsecutive convolutions. *arXiv* **2015**, arXiv:1508.04112.
21. Zhu, X.; Sobhani, P.; Guo, H. Long short-term memory over recursive structures. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1604–1612.

MDPI
St. Alban-Anlage 66
4052 Basel
Switzerland

Tel: +41 61 683 77 34
Fax: +41 61 302 89 18

www.mdpi.com

ISBN 978-3-0365-0231-1