Computers, Materials & Continua
DOI:10.32604/cmc.2021.016920
Article

Tech Science Press

Sentiment Analysis of Short Texts Based on Parallel DenseNet

Luqi Yan[1], Jin Han[1,*], Yishi Yue[2], Liu Zhang[2] and Yannan Qian[3]

[1]Nanjing University of Information Science & Technology, Nanjing, 210044, China
[2]State Grid Hunan Electric Power Company Limited Research Institute, Changsha, 410007, China
[3]Waterford Institute of Technology, Waterford, X91 K0EK, Ireland
[*]Corresponding Author: Jin Han. Email: hjhaohj@126.com
Received: 15 January 2021; Accepted: 04 March 2021

Abstract: Text sentiment analysis is a common problem in the field of natural language processing that is often resolved by using convolutional neural networks (CNNs). However, most of these CNN models focus only on learning local features while ignoring global features. In this paper, based on traditional densely connected convolutional networks (DenseNet), a parallel DenseNet is proposed to realize sentiment analysis of short texts. First, this paper proposes two novel feature extraction blocks that are based on DenseNet and a multi-scale convolutional neural network. Second, this paper solves the problem of ignoring global features in traditional CNN models by combining the original features with features extracted by the parallel feature extraction block, and then sending the combined features into the final classifier. Last, a model based on parallel DenseNet that is capable of simultaneously learning both local and global features of short texts and shows better performance on six different databases compared to other basic models is proposed.

Keywords: Sentiment analysis; short texts; parallel DenseNet

1 Introduction

With the development of computers and networks, people are increasingly using these networks to communicate. As shown in a recent survey, there were more than 904 million Internet users in China by April 2020, and the penetration rate of internet technology has reached 64.5% [1]. A variety of topics and comments on social media are spreading on the Internet, influencing every aspect of daily life. Applying natural language processing technology on social media has become an important way for enterprises to monitor public opinion, making it simple to analyze comments and understand the overall sentiment expressed by people.

Text sentiment analysis, also known as opinion mining, usually refers to dealing with short texts and deducing the overall sentiment expressed through modern information technology. There are three major methods for sentiment analysis of short texts at present, and they are based on a dictionary, traditional machine learning, and deep learning. The text sentiment analysis method based on a dictionary makes it possible for us to obtain the sentimental tendency of texts by counting and weighing the sentimental scores of texts according to the words with sentimental

information [2]. The text sentiment analysis method based on traditional machine learning does not depend on the dictionary and has the ability to learn the sentimental characteristics of texts by itself [2]. The text sentiment analysis method based on deep learning can learn more advanced and indescribable sentimental features of texts. Therefore, the features extracted by the text sentiment analysis method based on deep learning are abstract and difficult to express explicitly.

The popular text sentiment analysis model enables us to learn text expression features by using convolutional neural networks (CNNs) [3], recurrent neural networks (RNNs) [4] and graph convolutional neural networks (GCNs) [5]. The models of CNNs and RNNs can learn the local features of sentences better, while ignoring the global features, in that they give priority to location and order. The GCN model analyzes the relationship between words by constructing a graph CNN. Although the performance of long text analysis by GCN is excellent, the performance of short text analysis by GCN is not ideal. Therefore, this paper attempts to build a network to better adapt to the sentimental analysis of short texts.

In this paper, we propose a parallel DenseNet for sentiment analysis of short texts based on the traditional densely connected convolutional network (DenseNet). A novel defined convolutional feature extraction block is proposed that is different from the dense block proposed by Huang et al. [6]. First, each convolutional feature extraction block proposed will extract the features of the original text based on different feature extraction methods. Second, we will merge the output features of all convolutional feature extraction blocks with the original text. Finally, we will classify these output features using the classifier.

Briefly, the innovation of this paper is as follows: In this paper, a novel defined convolutional feature extraction block is proposed that can learn both the global and local features of texts, and is able to use different kernels to convolve the sentence to obtain features. Additionally, the network can obtain the long-distance dependency of the text by merging the output features of all convolutional feature extraction blocks. Compared with other CNN models, this model has a shorter convergence time and does not require multiple iterations of training.

The experimental results show that the proposed method is better than the latest text sentiment analysis method. The method developed in this paper has good performance in both small and large training datasets. In addition, the method is equally effective in the case of multi-classification, such as three classification, five classification, ten classification, and others.

The rest of this paper is organized as follows: Section 2 introduces the current situation of text sentiment analysis. Section 3 introduces the original definition of blocks. Section 4 introduces a parallel DenseNet for sentiment analysis of short texts. Section 5 introduces the data and schemes used in the experiment, and gives results of comparison and evaluation of the model's performance. Concluding remarks are given in Section 6.

2 Related Work

With regard to feature extraction in text sentiment analysis, there are three major methods: the bag-of-words model, the word embedding model, and the graph network model. The bag-of-words model is a very simple eigenvector representation model that has achieved many research results in text analysis tasks. The word embedding model is a model developed on the basis of the bag-of-words model that can contain more semantic information, and is the most important feature extraction method in text deep learning. The graph network model is a model developed in recent years that can analyze the sentiment of the text by constructing a network of the

relationships between words. As shown in Fig. 1, this section will summarize the text sentiment analysis according to the three major methods used in feature extraction.

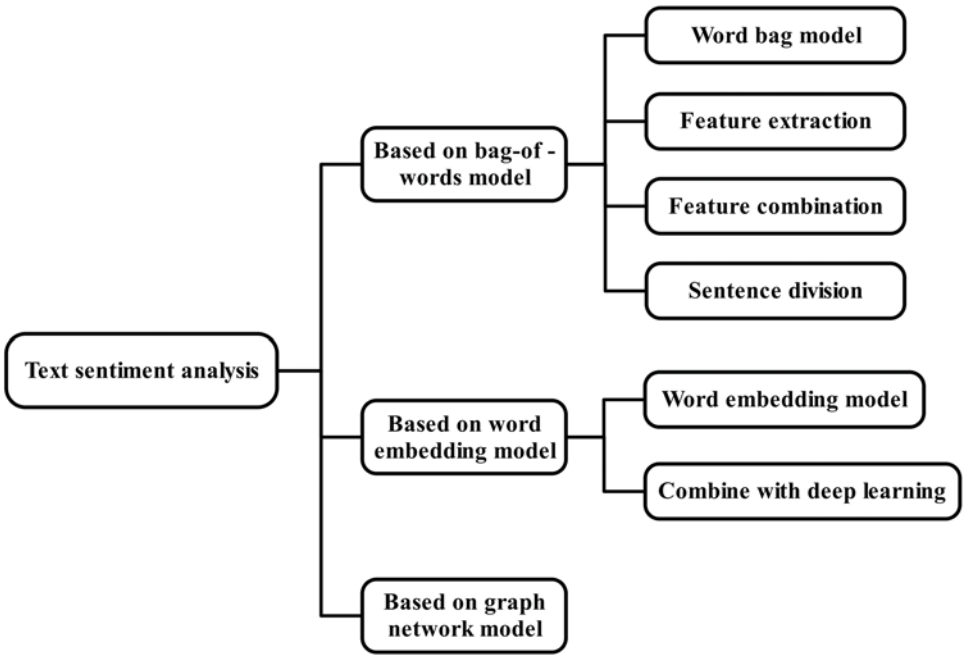

Figure 1: Research structure of text sentiment analysis

The principle of text sentiment analysis based on a bag-of-words model is to place all words into one bag, the so-called word bag. When a word appears in a sentence, the position of this word in the vector is 1, and the position of the other words is 0. In this case, the words in the sentence are out of order. Therefore, the bag-of-words model has been further developed into feature extraction methods, such as part-of-speech (POS) tagging and n-gram phrase tagging. Part-of-speech tagging, also known as grammatical tagging, is the process of marking words in a text (corpus) as corresponding to specific parts. N-gram phrase tagging is based on the fact that one word depends on several other words. When marking a word, that word is usually combined with the previous word. Chenlo et al. [7] and Priyanka et al. [8] combine POS and n-gram, and their experimental results show that this method can improve the classification accuracy. However, in the task of sentiment analysis of short texts, Kouloumpis et al. [9] found that this method could not achieve satisfactory accuracy. This is because these texts are short in length and similar to Weibo comments, and the composition of these sentences is extremely casual. Therefore, it is difficult to achieve satisfactory accuracy using part-of-speech tagging [10]. In terms of sentence division, Tang et al. [11] designed a classification framework that can identify words that lead to the transfer of sentimental polarity. Khan et al. [12] used the classification algorithm designed by SentiWordNet emotion score, and achieved a significant performance improvement on six evaluation datasets. SentiWordNet is a lexical resource for opinion mining that assigns to each synset of WordNet three sentiment scores: positivity, negativity, and objectivity. Some studies have shown that using SentiWordNet to query the sentiment value of a word and adding it as a feature can improve the accuracy of sentiment analysis [8,13]. The above description shows that the text classification based on a bag-of-words model can achieve better classification results when the word features

are properly obtained. However, the bag-of-words model also has some shortcomings, because it abandons the order between words, cannot convey deep-seated semantic features, and is unable to express semantic combination.

Text sentiment analysis based on a word embedding model solves the problem of the high-dimensional word vector in a bag-of-words model. The most frequently used word embedding model is the word2vec model. The word embedding model is based on the principle of "distance similarity" and has the function of smoothing. Another advantage of a word embedding model is that it is an unsupervised learning method. It has been proved that the word embedding model can obtain more semantic and grammatical features than the bag-of-words model [2]. This advantage enables the word embedding model to achieve very good results in a variety of natural language processing tasks. Tsvetkov et al. [14] designed a measurement method, QVEC, to evaluate the feature representation performance of various text analysis models. The experimental results show that for 300D word vectors, the QVEC score of the text sentiment analysis method based on a word embedding model is higher than that of other models. In recent years, more and more text analysis methods have adopted the combination of word embedding model and deep learning, and achieved better performance. Kombrink et al. [15] designed a word embedding learning algorithm that combines word vectors with an RNN and can be well applied to speech recognition. Cheng et al. [16] and Sundermeyer et al. [17] combine word vectors with long short-term memory (LSTM) to achieve better efficiency. Although the text CNN designed by Kim [3] has only one convolutional layer, its classification performance is significantly better than that of the ordinary machine learning classification algorithm. However, this method cannot obtain the long-distance dependencies of the text through convolution. Johnson et al. [18] extracted long-distance text dependencies by deepening the network and using residual ideas in 2017, but the performance was not satisfactory when the training data set was small. Wang et al. [19] introduced a structure similar to DenseNet in 2018 using a short-cut between the upper and lower convolutional blocks so that larger-scale features could be obtained from smaller-scale feature combinations. However, the model used a convolutional core of a specific size that slid from the beginning of the text to the end, producing a feature map. Yan et al. [20] introduced the method of small sample learning into text classification in order to solve the problem of poor text classification in the case of small sample size, and achieved good results, but the text classification is generally good in normal samples. Xiang et al. [21] and Yang et al. [22] designed a text steganography model by combining text with information hiding and achieved favorable results. Xiang et al. [23] achieved good results in spam detection by using LSTM-based multi-entity temporal features, but the results are generally good in sentiment analysis.

Text sentiment analysis based on graph network models improves on prior models by constructing a relationship network between words. In 2019 Yao et al. [5] classified texts by composing the unstructured data text through the co-occurrence information of words and articles, term frequency-inverse document frequency (TF-IDF) weight, and mutual information weight, and used GCN to capture the document–word, word–word, and document–document relationships in the graph. The experimental results showed that the classification performance of their model was excellent on long regular documents, but the classification effect and composition were not very ideal in short texts.

In this paper, we propose a parallel DenseNet for sentiment analysis of short texts. A novel defined convolutional feature extraction block is designed in this network. Like the network designed by Kim [3], the convolutional feature extraction block can use different kernels to convolve sentences in one dimension to obtain features. Furthermore, the network can obtain the

long-distance dependency of the text by merging the output features of all convolutional feature extraction blocks. The experimental results show that this method has the same efficiency in both small and large training datasets.

3 Background

In this section, we briefly introduce the definition and components of the dense block proposed from Huang et al. [6] in 2017. Next, we will briefly introduce the difference between the block in this paper and the block proposed by Huang.

As shown in Fig. 2, the input of each layer in the dense block proposed by Huang is the concatenation of the outputs of all previous layers. At the same time, all dense blocks have the same structure in DenseNet, which means that the number of internal layers and the size of the convolutional kernels are exactly the same.

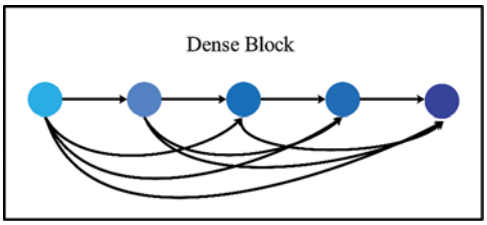

Figure 2: The structure of the dense block proposed in 2017

The block proposed in this paper, which is called the convolutional feature extraction block, has a unique internal structure. For example, the two feature extraction blocks, consisting of a densely connected convolutional feature extraction block and a multi-scale convolutional feature extraction block, have completely different internal structures. The densely connected convolutional feature extraction block is similar to the dense block proposed by Huang. The input to each layer in the block comes from the sum of the outputs of all previous layers. The multi-scale convolutional feature extraction block is completely different from the dense block proposed by Huang. There is a parallel relationship between the layers in the block, and each layer uses a different window size similar to the n-gram method for feature extraction. In this paper, an independent feature extraction block can be called a block and does not necessarily need to have the structure of the dense block proposed by Huang.

4 Method

4.1 Overview

Our goal is to improve the performance of short text classification through a parallel DenseNet. The overall model of this paper is shown in Fig. 3. At the beginning of the model, a text x of length m is entered. Here, $x = [x_1, x_2, \ldots, x_m]_{m \times d}$, $x_i \in \mathbb{R}^d$. Then, the text is input into two convolutional feature extraction blocks, which are a densely connected convolutional feature extraction block and a multi-scale convolutional feature extraction block, and the features are extracted in parallel. Then, the total feature $\mathcal{X}$ is obtained by combining the features $\mathcal{X}_1$ and $\mathcal{X}_2$ extracted from two convolutional blocks with the feature $\mathcal{X}_3$ extracted from the text through the maximum pool block with a size of 50. Here, $\mathcal{X}_1 \in \mathbb{R}^{k \times n_1}$, $\mathcal{X}_2 \in \mathbb{R}^{k \times n_2}$, $\mathcal{X}_3 \in \mathbb{R}^{k \times n_3}$, $\mathcal{X} \in \mathbb{R}^{k \times n}$,

and $n = n_1 + n_2 + n_3$. Finally, the total feature is pooled through the global average to obtain the average total feature $\widetilde{\mathcal{X}}$, and then classified according to the average total feature. Here, $\widetilde{\mathcal{X}} \in \mathbb{R}^n$.

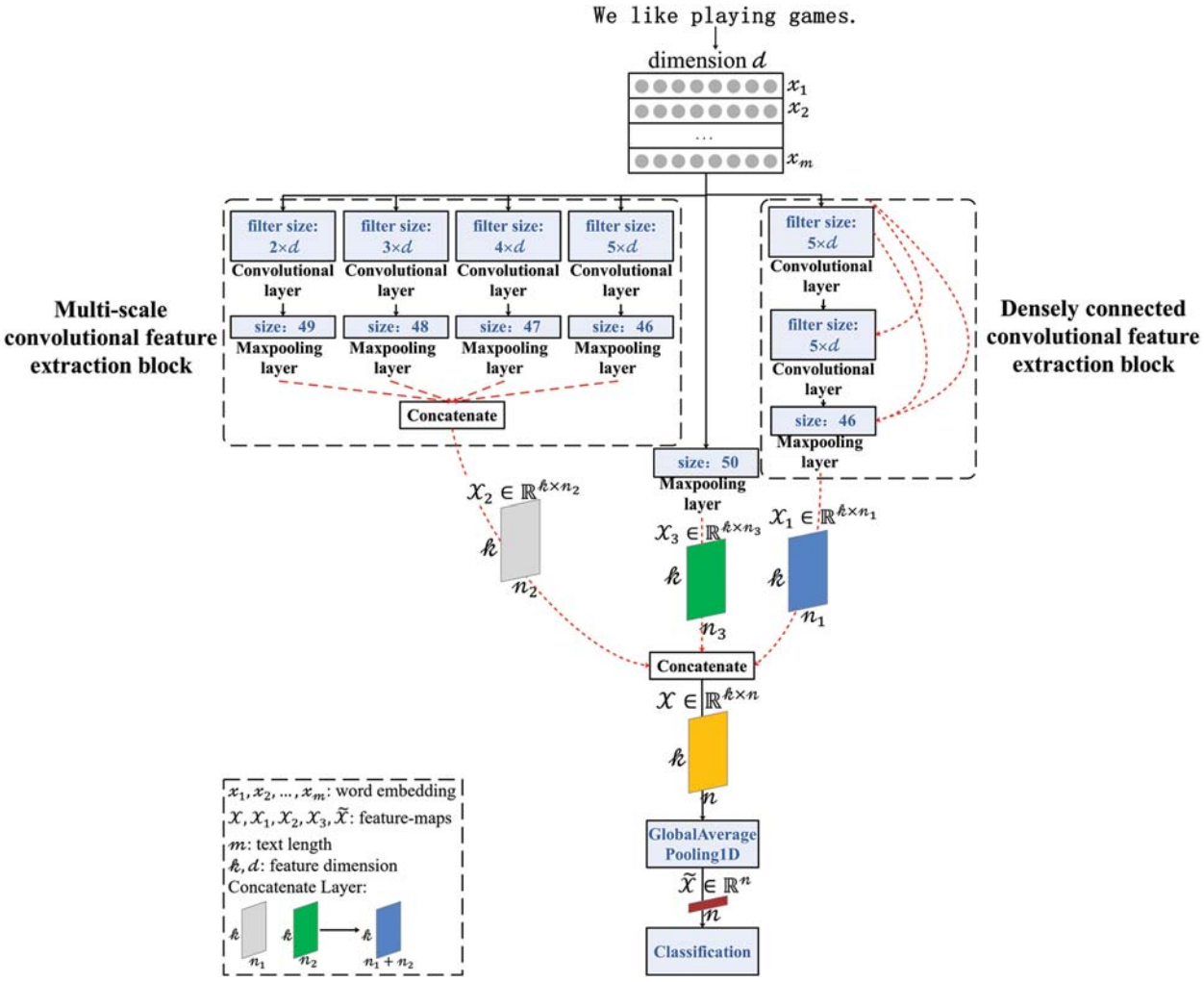

Figure 3: Framework of parallel densely connected convolutional neural network

4.2 Densely Connected Convolutional Feature Extraction Block

As shown in Fig. 4, let $x_i \in \mathbb{R}^d$ be the d-dimensional pre-trained word vector of the i-th word in the text, then the original input text can be represented as a matrix.

$$x^0 = [x_1, x_2, \ldots, x_m]_{m \times d} \tag{1}$$

Here, m is the number of words in the text, and d is the dimension that each word is pre-trained into a word vector.

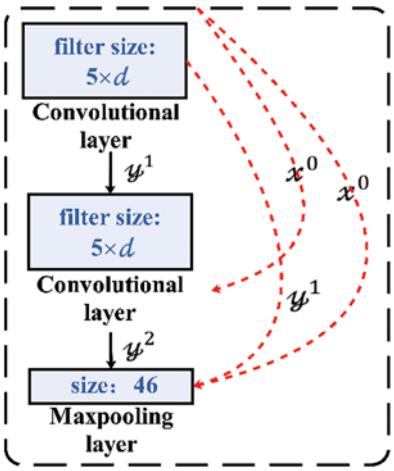

Figure 4: The model of the densely connected convolutional feature extraction block

Then, the original input text matrix is input into a convolutional layer with a size of $5 \times d$ for feature extraction (using a combination of five words for feature extraction).

$$y^1 = f_{5\times d}\left(x^0\right) \tag{2}$$

Here, x^0 is the original input text matrix, $f_{5\times d}$ is the convolutional transformation with a convolutional kernel size of $5 \times d$, and y^1 is the characteristic matrix after a convolutional transformation.

Then, the original input text matrix is combined with the characteristic matrix after a convolutional transformation, and a new input text matrix is obtained.

$$x^1 = Cat\left(\left[x^0, y^1\right]\right) \tag{3}$$

Here, x^0 is the original input text matrix, y^1 is the characteristic matrix after a convolutional transformation, and *Cat* refers to the splicing and merging of multiple matrices in the last dimension.

Then the new input text matrix is input into a convolutional layer with a size of $5 \times d$ for feature extraction (using a combination of five words for feature extraction).

$$y^2 = f_{5\times d}\left(x^1\right) \tag{4}$$

Here, x^1 is a new input text matrix, y^2 is a characteristic matrix after quadratic convolutional transformation, and $f_{5\times d}$ is a convolutional transformation with a convolutional kernel size of $5 \times d$.

Then the original input text matrix, the characteristic matrix after primary convolutional transformation and the characteristic matrix after quadratic convolutional transformation are combined to obtain a new feature matrix.

$$x^2 = Cat\left(\left[x^0, y^1, y^2\right]\right) \tag{5}$$

Here, x^0 is the original input text matrix, y^1 is the characteristic matrix after a convolutional transformation, y^2 is the characteristic matrix after a convolutional transformation, and *Cat* refers to the splicing and merging of multiple matrices in the last dimension.

Finally, the new eigenmatrix is input into the maximum pool layer with a size of 46, and the eigenmatrix of the densely connected convolutional feature extraction block is obtained.

$$\mathcal{X}_1 = h_{46}\left(x^2\right) \tag{6}$$

Here, x^2 is the new eigenmatrix, and h_{46} is the maximum pool transformation of size 46.

4.3 Multi-Scale Convolutional Feature Extraction Block

As shown in Fig. 5, in accordance with the densely connected convolutional feature extraction block module, let $x_i \in \mathbb{R}^d$ be the d-dimensional pre-training word vector x^0 of the i-th word in the text. Then, the original input text matrix is input to convolutional layers with sizes $5 \times d$, $4 \times d$, $3 \times d$, and $2 \times d$ at the same time for feature extraction (using a combination of words, i.e., five, four, three, and two words for feature extraction).

$$\begin{aligned} y^1 &= f_{5\times d}\left(x^0\right) \\ y^2 &= f_{4\times d}\left(x^0\right) \\ y^3 &= f_{3\times d}\left(x^0\right) \\ y^4 &= f_{2\times d}\left(x^0\right) \end{aligned} \tag{7}$$

Here, x^0 is the original input text matrix and y^1, y^2, y^3, and y^4 represent the characteristic matrix after convolutional transformation with convolutional kernel sizes of $5 \times d$, $4 \times d$, $3 \times d$, and $2 \times d$, respectively.

Then, after the convolutional transformation of the convolutional kernel size of $5 \times d$, $4 \times d$, $3 \times d$, and $2 \times d$, the maximum pool operation of the maximum pool layer with the input sizes of 46, 47, 48, and 49, respectively, is carried out to obtain a new characteristic matrix.

$$\begin{aligned} x^1 &= h_{46}\left(y^1\right) \\ x^2 &= h_{47}\left(y^2\right) \\ x^3 &= h_{48}\left(y^3\right) \\ x^4 &= h_{49}\left(y^4\right) \end{aligned} \tag{8}$$

Here, y^1, y^2, y^3, and y^4 represent the eigenmatrix after convolutional transformation with convolutional kernel sizes of $5 \times d$, $4 \times d$, $3 \times d$, and $2 \times d$; h_{46}, h_{47}, h_{48}, and h_{49} are the maximum pool transformations of input size 46, 47, 48, and 49; and x^1, x^2, x^3, and x^4 are the new eigenmatrices.

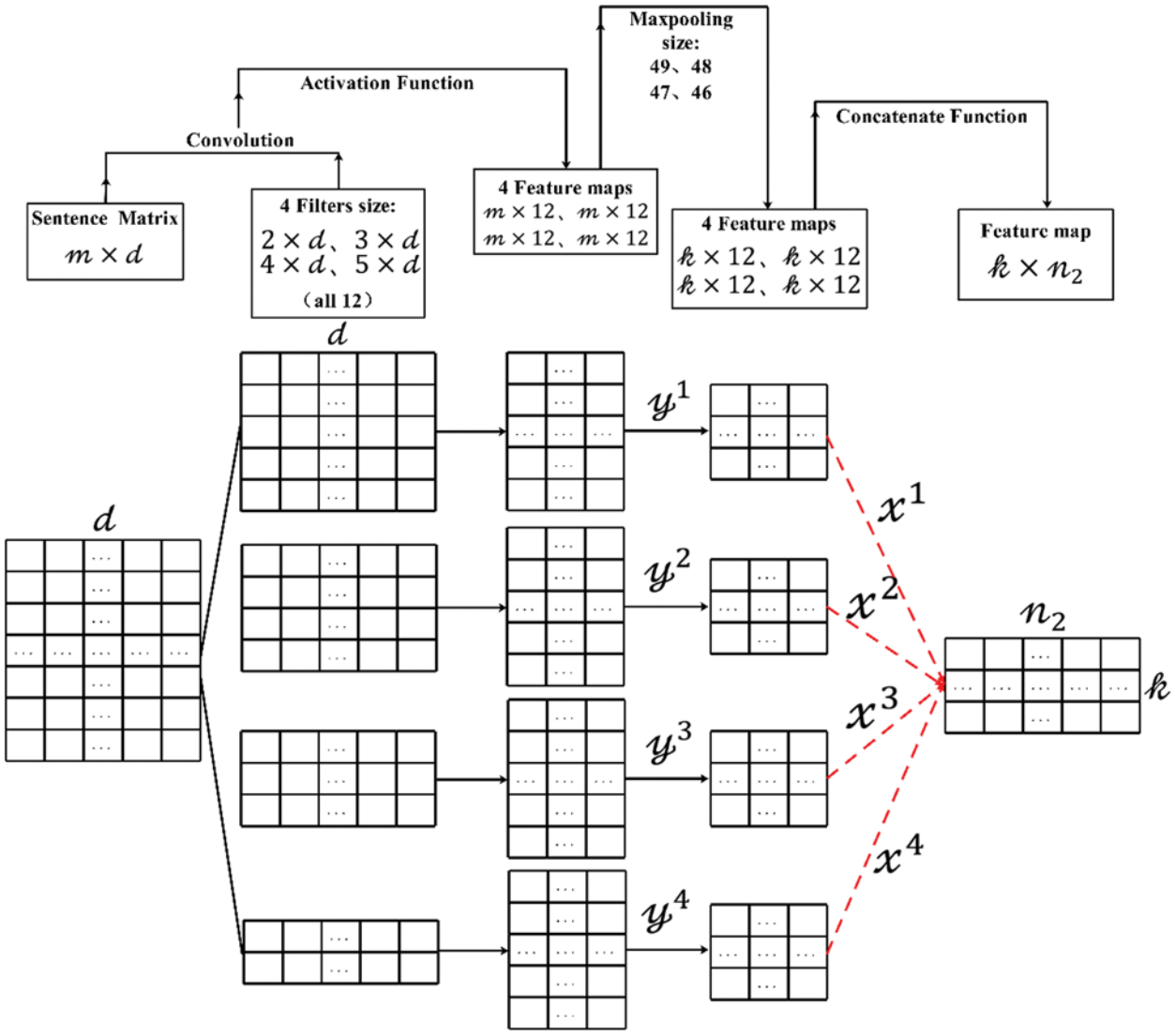

Figure 5: The model of the multi-scale convolutional feature extraction block

Finally, the new eigenmatrices are combined to obtain the eigenmatrix of the multi-scale convolutional feature extraction block.

$$\mathcal{X}_2 = Cat\left(\left[x^1, x^2, x^3, x^4\right]\right) \tag{9}$$

Here, *Cat* refers to the splicing and merging of multiple matrices in the last dimension, and x^1, x^2, x^3, and x^4 are the new characteristic matrices.

4.4 Text Classification

As shown in Fig. 6, the total feature matrix $\mathcal{X}$ is obtained by combining the feature matrices $\mathcal{X}_1$ and $\mathcal{X}_2$ obtained from the parallel feature extraction of the original input text matrix through the above two convolutional feature extraction blocks, consisting of a densely connected

convolutional feature extraction block and a multi-scale convolutional feature extraction block, and the feature matrix $\mathcal{X}_3$, which is extracted from the original input text matrix through the largest pool sblock with a size of 50.

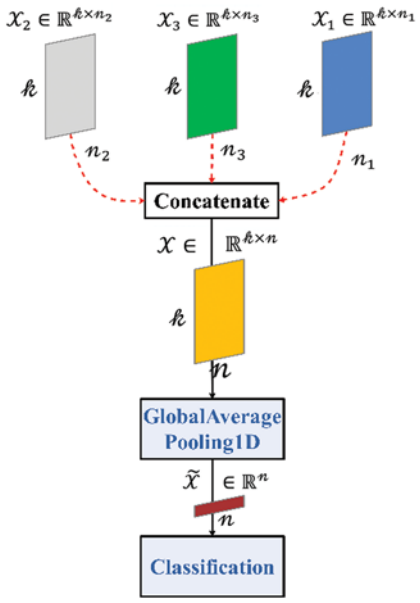

Figure 6: The model of text classification

Here, $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{X}_3$ represent the feature matrices obtained by the above feature extraction, and *Cat* refers to the stitching and merging of multiple matrices in the last dimension.

Then, $\mathcal{X}$ is pooled by one-dimensional global averaging to obtain the final eigenmatrix.

$$\tilde{\mathcal{X}} = g(\mathcal{X}) \tag{10}$$

Here, $\mathcal{X}$ denotes the total eigenmatrix, $\tilde{\mathcal{X}}$ represents the final eigenmatrix obtained by one-dimensional global average pooling of $\mathcal{X}$, and g represents one-dimensional average pooling.

Finally, the final feature matrix is input into the classification layer for text classification.

5 Experiments

In this section, the model of the experiment is introduced in detail, and the results of the experiment are analyzed and discussed.

5.1 Experimental Setup

To verify the rationality and validity of the model, six widely used benchmark corpora were selected and tested. These include the GameMultiTweet dataset, SemEval dataset, SS-Tweet dataset, AG News dataset, R8 dataset, and Yahoo! Answers dataset.

- The GameMultiTweet dataset is built by searching game data and other game themes. In this dataset, 12780 pieces of data are separated into three categories, and the proportion of categories is 3952:915:7913.

- The SemEval dataset consists of 20K data created by the Twitter sentiment analysis task. In this dataset, 7967 pieces of data are separated into three categories, and the proportion of categories is 2964:1151:3852.
- The SS-Tweet dataset is the sentimental intensity Twitter dataset. In this dataset, 4242 pieces of data are separated into three categories, and the proportion of categories is 1953:1336:953.
- The AG News dataset is a collection of more than 1 million news articles from more than 2000 different news sources after more than a year of efforts by ComeToMyHead. In this dataset, 127600 pieces of data are separated into four categories, and the proportion of categories is 31900:31900:31900:31900.
- The R8 dataset is a collection of approximately 20000 newsgroup documents. In this dataset, 4203 pieces of data are separated into eight categories, and the proportion of categories is 1392:241:2166:20:162:0:72:150.
- The Yahoo! Answers dataset is the 10 main classification data of the Yahoo! Answers Comprehensive Questions and Answers 1.0 dataset. In this dataset, 350000 pieces of data which are separated into 10 categories, and the proportion of categories is 23726:35447:31492:35252:35546:25787:81571:23961:28706:28482.

All datasets were randomly divided into the following three parts: 70% training set, 15% verification set, and 15% test set. The specific dataset statistics are shown in Tab. 1 below.

Table 1: Summary statistics of datasets

Dataset	#Train	#Validation	#Test	Categories	Avg. length
GameMultiTweet	8964	1917	1917	3	26
SemEval	5577	1195	1195	3	31
SS-Tweet	2970	636	636	3	29
AG news	89320	19140	19140	4	45
R8	2943	630	630	8	66
Yahoo! answers	245000	52500	52500	10	112

In this paper, the novel method is compared with the following benchmark models:

- CNN: A CNN model composed of three layers of one-dimensional convolutional layers, with convolutional kernels of each layer being the same size.
- TextCNN: A method proposed by Kim [3] in 2014 that applies CNN to text classification tasks. This method extracts the key information from the text according to the convolutional kernels of different sizes (the function of the convolutional kernels of different sizes is similar to the n-gram of different sizes), so as to better obtain the local features of the text.
- FastText: A simple and efficient text classification method proposed by Joulin et al. [24] in 2017. The core idea of this method is to obtain the text vector by averaging the word vector of the whole text and the vector superimposed by n-gram vector, and then the vector is multi-classified by softmax.
- DPCNN: A deep CNN proposed by Johnson et al. [18] in 2017. The core idea of this method is to take the word vector of each word of the text as input and extract features through the network to achieve the purpose of classification. Each convolutional block in

the network consists of two convolutional layers. The connection between the convolutional blocks is made by jumping, and the input of each convolutional block is the result of the addition of the output of the previous convolutional block and the identity mapping. The sampling block is downsampled with a scale of 2 to achieve the purpose of scaling. Several convolutional blocks and sampling blocks are stacked to form a scale pyramid to achieve the purpose of dimension scaling. Finally, the output is spliced into vectors through the hidden layer and softmax layer as the output classification.

5.2 Implementation Details

In this study, the dimension of each word in the text was set to 300 dimensions, and the maximum number of words in each sentence was set to 150. Each sentence was transformed into a "150×300" matrix by word2vector. Some parameters were set, such as using the adam optimizer [25], and setting the learning rate to 0.001, dropout rate to 0.2, and L2 loss weight to 10^{-8}. The model batch size was 50 and the number of epochs was 5. If the loss was not reduced in 10 consecutive periods, the training was stopped.

For the benchmark model, the parameter settings we used were the same as those set in the original article. In the pre-training word embedding model, 300D word2vector word embedding was used.

5.3 Experimental Results

Tab. 2 shows the results of the model and the benchmark model in this paper. From the results, we can see that the model in this paper can achieve better accuracy than its competitors.

Table 2: Test accuracy on several text classification datasets

Model	GameMultiTweet	SemEval	SS-Tweet	AG news	R8	Yahoo! answers
CNN	73.5 ± 0.5	60.5 ± 0.3	50.2 ± 0.5	85.6 ± 0.6	92.3 ± 0.3	47.3 ± 0.5
TextCNN	77.5 ± 0.6	62.7 ± 0.5	51.1 ± 0.8	88.9 ± 0.5	94.4 ± 0.5	49.5 ± 1.0
FastText	78.3 ± 0.3	63.8 ± 0.2	51.4 ± 1.0	88.5 ± 1.0	96.1 ± 0.2	49.8 ± 1.0
DPCNN	75.6 ± 1.0	47.5 ± 1.0	43.2 ± 1.0	87.1 ± 1.0	88.5 ± 1.0	47.5 ± 1.0
Our model	78.5 ± 0.8	66.0 ± 0.6	52.4 ± 0.5	89.7 ± 0.6	98.1 ± 0.8	51.6 ± 0.8

As can be seen from the results, based on large datasets (AG News and Yahoo! Answers) and small datasets (GameMultiTweet, SemEval, SS-Tweet and R8), the model in this paper is more accurate than traditional models, such as CNN, TextCNN, FastText, and DPCNN. Both the model in this paper and the benchmark model choose filter stop words and part-of-speech tagging in feature extraction. Although TextCNN contains only one layer of convolutional operation in the model, it is much better than CNN with three layers and one-dimensional convolutional layer in the task of text classification. Therefore, the text features extracted by convolutional kernels of different sizes in TextCNN can better reflect the local features of the text, which is more conducive to the task of text classification. Although FastText is a very simple linear model that takes the average value of word vector and n-gram vector as its text feature vector, it is better than TextCNN, which has a layer convolutional operation to obtain multi-scale maximum feature vector combination in a text classification task. Therefore, the average value of the superposition of multi-scale vector features and word vectors can better extract the global features of the text. DPCNN uses the jump connection between convolutional blocks and the sampling block to scale

down with the size of 2 to achieve the purpose of obtaining long-distance features. Although the performance is excellent in the original article, the performance is not ideal on the experimental data set of this paper. The model in this paper combines the advantages of TextCNN and FastText. Local features can be extracted by convolutional kernels of different sizes, and global features can be obtained by averaging. At the same time, compared with other deep CNN models, the model can converge with very few epochs and does not need multiple iterative training.

The number of samples has an obvious effect on the performance of the model. For SemEval, SS-Tweet, and R8 datasets, because of the small sample size of these three datasets, the accuracy of the DPCNN model is significantly different from that of other models. Therefore, it can be seen that deep neural network models such as DPCNN are not effective in the task of text classification. However, although the model presented in this paper also utilizes a deep neural network, it has a better classification effect on the small sample dataset.

The length of the sample text has an obvious effect on the performance of the model. For SS-Tweet and R8 datasets, because the sample size of the two datasets is similar and the average text length is not the same, the classification accuracy of the two datasets is very different. However, the classification effect of this model is the best on these two datasets.

5.4 Tuning of Hyperparameters

Epoch size: Through experiments, this paper shows the influence of epoch size on the size of the model. In this paper, epoch size was parameterized within the range {3, 4, 5, 6, 7}. As shown in Fig. 7a, epoch size has a great influence on the model. Therefore, we adjusted epoch size several times before converging on the best result.

Network depth: This paper assesses how the depth of the network affects performance by changing the size of the densely connected convolutional feature extraction block. In this paper, network depth is parameterized within the range {1, 2, 3, 4, 5}. As shown in Fig. 7b, network depth has a small impact on the model. Although the increase of network depth in densely connected convolutional feature extraction block will make the features extracted by the block more obvious, the improvement is not obvious for the whole model.

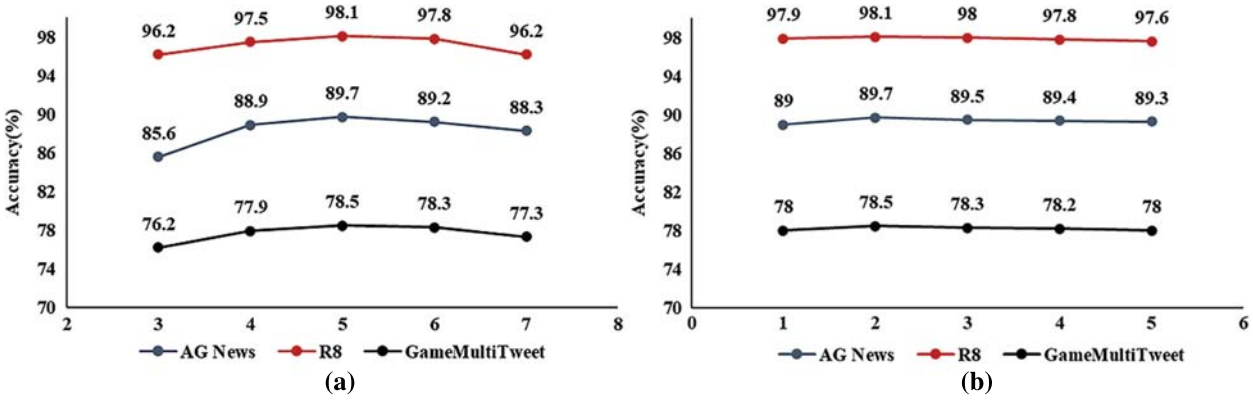

Figure 7: Accuracy with different epoch sizes and network depths. (a) Epoch size. (b) network depth

6 Conclusion

In this paper, we propose a parallel DenseNet for sentiment analysis of short texts, in which a novel convolutional feature extraction block is defined. This model extracts features by using convolutional feature extraction blocks and then conducts feature extraction and classification by merging these features with original text features. Compared with other deep CNN models, this model has a smaller convergence time and does not require multiple iterations of training. The model demonstrates competitive performance on six datasets. Our analysis reveals that this model can extract both global features and local features, and obtain best performance when compared to its peers.

Funding Statement: This work was supported by the National Key R&D Program of China under Grant Number 2018YFB1003205; by the National Natural Science Foundation of China under Grant Numbers U1836208, U1536206, U1836110, 61602253, and 61672294; by the Startup Foundation for Introducing Talent of NUIST (1441102001002); by the Jiangsu Basic Research Programs-Natural Science Foundation under Grant Number BK20181407; by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund; and by the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET) fund, China.

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

References

[1] China Internet Research Center, *The 45th "Statistical Report on Internet Development in China*." Beijing, China: China Internet Network Information Center, 2020. [Online]. Available: http://www.cac.gov.cn/2020-04/27/c_1589535470378587.htm.

[2] Y. X. Yu, "Research and optimization of text sentiment analysis based on machine learning," M.S. dissertation, Beijing University of Posts and Telecommunications, Beijing, 2018.

[3] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, Doha, Qatar, pp. 1746–1751, 2014.

[4] P. Liu, X. Qiu and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proc. IJCAI*, New York, NY, USA, pp. 2873–2879, 2016.

[5] L. Yao, C. Mao and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI*, Hawaii, USA, pp. 7370–7377, 2019.

[6] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, Hawaii, USA, pp. 4700–4708, 2017.

[7] J. M. Chenlo and D. E. Losada, "An empirical study of sentence features for subjectivity and polarity classification," *Information Sciences*, vol. 280, pp. 275–288, 2014.

[8] C. Priyanka and D. Gupta, "Identifying the best feature combination for sentiment analysis of customer reviews," in *Proc. ICACCI*, Mysore, India, pp. 102–108, 2013.

[9] E. Kouloumpis, T. Wilson and J. Moore, "Twitter sentiment analysis: The good the bad and the omg!," in *Proc. ICWSM*, Barcelona, Spain, pp. 538–541, 2011.

[10] S. Sun, H. Liu and A. Abraham, "Twitter part-of-speech tagging using pre-classification Hidden Markov model," in *Proc. IEEE SMC*, Seoul, South Korea, pp. 1118–1123, 2012.

[11] D. Tang, F. Wei, B. Qin, L. Dong, T. Liu *et al.,* "A joint segmentation and classification framework for sentiment analysis," in *Proc. EMNLP*, Doha, Qatar, pp. 477–487, 2014.

[12] F. H. Khan, S. Bashir and U. Qamar, "TOM: Twitter opinion mining framework using hybrid classification scheme," *Decision Support Systems*, vol. 57, pp. 245–257, 2014.

[13] W. Chamlertwat, P. Bhattarakosol, T. Rungkasiri and C. Haruechaiyasak, "Discovering consumer insight from twitter via sentiment analysis," *Journal of Universal Computer Science*, vol. 18, no. 8, pp. 973–992, 2012.

[14] Y. Tsvetkov, M. Faruqui, W. Ling, G. Lample and C. Dyer, "Evaluation of word vector representations by subspace alignment," in *Proc. EMNLP*, Lisbon, Portugal, pp. 2049–2054, 2015.

[15] S. Kombrink, T. Mikolov, M. Karafiát and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *Proc. INTERSPEECH*, Florence, Italy, pp. 2877–2880, 2011.

[16] J. Cheng, X. Zhang, P. Li, S. Zhang, Z. Ding *et al.,* "Exploring sentiment parsing of microblogging texts for opinion polling on chinese public figures," *Applied Intelligence*, vol. 45, no. 2, pp. 429–442, 2016.

[17] M. Sundermeyer, R. Schlüter and H. Ney, "LSTM neural networks for language modeling," in *Proc. INTERSPEECH*, Portland, OR, USA, pp. 194–197, 2012.

[18] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proc. ACL*, Vancouver, Canada, pp. 562–570, 2017.

[19] S. Wang, M. Huang and Z. Deng, "Densely connected CNN with multi-scale feature attention for text classification," in *Proc. IJCAI*, Stockholm, Sweden, pp. 4468–4474, 2018.

[20] L. Yan, Y. H. Zheng and J. Cao, "Few-shot learning for short text classification," *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29799–29810, 2018.

[21] L. Xiang, S. Yang, Y. Liu, Q. Li and C. Zhu, "Novel linguistic steganography based on character-level text generation," *Mathematics*, vol. 8, no. 9, pp. 1558, 2020.

[22] Z. Yang, S. Zhang, Y. Hu, Z. Hu and Y. Huang, "VAE-Stega: Linguistic steganography based on variational auto-encoder," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 880–895, 2021.

[23] L. Xiang, G. Guo, Q. Li, C. Zhu, J. Chen *et al.,* "Spam detection in reviews using lstm-based multi-entity temporal features," *Intelligent Automation & Soft Computing*, vol. 26, no. 6, pp. 1375–1390, 2020.

[24] A. Joulin, É. Grave, P. Bojanowski and T. Mikolov, "Bag of tricks for efficient text classification," in *Proc. EACL*, Valencia, Spain, pp. 427–431, 2017.

[25] D. P. Kingma and L. J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, San Diego, CA, 2015.