# CNN-ELM Hybrid Short Text Classification Model Postprint

**Authors:** Han Zhonghe, Xia Dynasty and Warring States Period, Yang Ting

**Date:** 2018-05-18T00:00:00+00:00

## Abstract

In current natural language processing research, to address the issue that convolutional neural networks (CNN) for short text classification tasks can be combined with different neural network structures and classification algorithms to improve classification performance, we propose a CNN-ELM hybrid short text classification model that combines convolutional neural networks with extreme learning machines. Word vectors are trained to form a text matrix as input data, convolutional neural networks are then employed to extract features and Highway networks are utilized for feature optimization, and finally error-minimized extreme learning machines (EM-ELM) serve as the classifier to complete the short text classification task. Compared with other models, this hybrid model can extract more representative features and output classification results quickly and accurately. Experimental results on multiple English datasets demonstrate that the proposed CNN-ELM hybrid short text classification model is more suitable for short text classification tasks than traditional machine learning models and deep learning models.

## Full Text

## Preamble

### Hybrid CNN-ELM Model for Short Text Classification

Han Zhonghe[1], Xia Zhanguo[1], Yang Ting[2]
(1. College of Computer Science & Technology, China University of Mining & Technology, Xuzhou, Jiangsu 221000, China;
2. Suzhou Institute for Advanced Study, Institute of Electronics, Chinese Academy of Sciences, Suzhou, Jiangsu 215000, China)

**Abstract:** In current natural language processing research, combining different neural network structures with classification algorithms when using Convolutional Neural Networks (CNN) for short text classification tasks can im-

prove classification performance. This paper proposes a hybrid CNN-ELM short text classification model that integrates CNN with Extreme Learning Machine (ELM). The model uses word vector training to construct text matrices as input data, employs CNN for feature extraction, utilizes Highway networks for feature optimization, and finally uses Error-Minimized Extreme Learning Machine (EM-ELM) as the classifier to complete short text classification tasks. Compared with other models, this hybrid model can extract more representative features and output classification results quickly and accurately. Experimental results on multiple English datasets demonstrate that the proposed CNN-ELM hybrid short text classification model is more suitable for short text classification tasks than traditional machine learning models and deep learning models.

**Keywords:** text classification; convolutional neural networks; extreme learning machine

## 0 Introduction

Deep learning has become increasingly prevalent in natural language processing (NLP) applications in recent years, with short text classification representing a particularly important component. Short text classification refers to the process of categorizing valuable short text information, which holds significant importance in today' s information society. The most critical challenge in short text classification lies in text feature extraction. Traditional feature extraction methods such as MI [1], pLSA [2], and LDA [3] often overlook contextual relationships within text, thereby failing to accurately capture lexical semantics.

In recent years, the remarkable performance of deep learning in image recognition and handwriting recognition has been widely acknowledged. However, applying deep learning to text processing requires digital representation of text. Word embedding has emerged as the most effective approach for preserving lexical grammatical and semantic information. This algorithmic training process expresses lexical similarity as vector space similarity, substantially retaining semantic and contextual information [4,5]. With the aid of word embeddings, applying deep learning to extract rich features from text becomes feasible.

Convolutional Neural Networks (CNN), as one of the most representative deep learning architectures, have been extensively applied in text processing. Among these CNN structures, the model proposed by Kim in 2014 [6] effectively demonstrated the substantial impact of applying word vectors to simple CNN structures for text classification, making it one of the most representative models for CNN applications in NLP. Subsequently, Mandelbaum et al. [7] implemented and improved Kim' s model on TensorFlow, further enhancing classification accuracy across multiple English datasets. In domestic research, Chen et al. [8] combined binary features from sentiment dictionary recognition with CNN-extracted features, and this method of adding external auxiliary features significantly improved the sentiment analysis capability of CNN models. Liu et al. [9] demonstrated that using character-level features for CNN sentiment analy-

sis is more effective than word-level features. Tong et al. [10] combined different channel word vectors as CNN input for entity relation extraction, achieving higher macro-averaged F1 scores than CNN and RNN models. These methods primarily focus on improving CNN structure inputs and feature representation. Beyond these approaches, combining CNN feature extraction models with different classification principles can also effectively enhance model performance.

Currently, the most common approach for combining CNN with different classification principles involves integrating CNN with SVM, which has been applied in sentiment analysis and face recognition, yielding better results than traditional CNN classification models [11,12]. However, when using cross-validation to partition training and test sets, SVM incurs significant time costs for parameter determination, while its performance still has room for improvement.

Huang et al. proposed Extreme Learning Machine (ELM) [13], a powerful machine learning model that is a single-hidden layer feedforward neural network (SLFN) capable of randomly selecting hidden nodes and calculating output weights. ELM features strong generalization ability and extremely fast learning speed, with studies indicating that ELM classifiers outperform SVM classifiers [14]. In recent years, methods combining extreme learning machines with convolutional features have been implemented in specific domains [15], though no research has yet applied this approach to text processing. While ELM possesses excellent generalization capabilities, obtaining optimal results typically requires manual statistical methods. As an improvement to ELM, Error-Minimized Extreme Learning Machine (EM-ELM) [16] enables automatic calculation of optimal solutions and continuous updating of network output weights.

Inspired by LSTM networks, Srivastava et al. [17] proposed Highway networks in 2015. By performing feature optimization similar to gating mechanisms on features extracted by deep neural networks, this architecture effectively addresses the difficulty of converging during training of deep multi-layer networks. In practical applications, R.K. Srivastava et al. [18] combined this new network structure with CNN and fully connected networks for experiments on image recognition datasets (CIFAR, MNIST, etc.), with results demonstrating that deep neural networks integrated with Highway networks achieved higher accuracy. Beyond image processing, Kim applied Highway networks to natural language processing, proposing a model that combines CNN with Highway networks and LSTM using character-level input for multilingual language analysis tasks [19]. He also analyzed the importance of Highway networks in this model, with experiments on the PTB dataset showing that Highway networks can optimize features extracted by CNN and that performance is optimal with 2 Highway network layers. In speech recognition applications, Wei-Ning Hsu et al. [20] added Highway networks to LSTM within a CLDNN model composed of CNN, LSTM, and DNN, achieving state-of-the-art performance on Chinese broadcast speech datasets.

In summary, this paper proposes a CNN-ELM hybrid short text classification model. Unlike the combination of CNN and ELM for image classification [15],

our model uses one-dimensional convolution to obtain feature vectors combined with multi-layer Highway networks to form a deep network. Additionally, improvements to ELM are not limited to parameter initialization optimization [15], as such approaches still require extensive hidden node testing to obtain optimal results. Our model uses EM-ELM to directly output optimal results, avoiding the time consumption associated with extensive testing.

Experimental results demonstrate that the proposed model outperforms traditional machine learning classification models and conventional CNN models on multiple English short text classification datasets. The innovations of this paper include: (1) Combining convolutional neural networks with extreme learning machine principles to improve model generalization capability and effectively enhance classification performance; (2) Adding Highway network layers for feature optimization in the hybrid model to further improve performance, while also investigating the impact of different Highway network layer numbers on classification performance.

## 1 Data Preprocessing and Word Vector Representation

### 1.1 Data Preprocessing

Since this study uses English short text datasets, the linguistic expressions in these short texts possess relatively special properties of online language, presenting numerous challenges for short text classification tasks. During data preprocessing, we employ standard text cleaning techniques to remove punctuation marks and irrelevant symbols, thereby reducing the workload for word vector conversion and classification. Considering that short text datasets contain numerous complex attributes such as web addresses, symbols, and emoticons, we use regular expressions to identify URL markers (http), @ symbols, topic tags (#), and simple emoticons, converting non-English text into feature markers. Finally, all unrecognized symbols and words are categorized as out-of-vocabulary cases and converted to random-valued word vectors during word vector training.

### 1.2 Word Vector Representation

Word vectors have various definitions, with the most widely accepted being that a word vector is a digital representation of a word, typically in vector form. More precisely, word vectors represent a technique that associates the semantics of a word with a vector obtained through unsupervised training methods. Unlike images, which are inherently rich, high-dimensional vectors, textual language cannot directly serve as input data for data mining algorithms for deeper analysis, making its discrete representation extremely important.

Initially, One-hot word vectors were used to represent vocabulary, employing a vector for each word with length equal to dictionary size, containing a single 1 component and all other components as 0, where the position of component 1 corresponds to the word' s position in the dictionary. While simple, this method

has obvious drawbacks: (a) it suffers from the curse of dimensionality, often resulting in extremely sparse word vectors for short text data, particularly when applied to certain algorithms in deep learning; (b) it cannot effectively capture similarity between words, exhibiting the "lexical gap" phenomenon where any two words are isolated, and their relationship cannot be determined from their vectors alone.

To address One-hot vector limitations, Hinton proposed word embedding, which distributes words into a low-dimensional space, solving vector sparsity issues. Furthermore, the positional relationships between word vectors in this low-dimensional space effectively reflect their semantic connections, making them highly suitable as high-level abstract features for text.

Currently, most researchers use Word2Vec, a software tool released by Google in 2013 for training word vectors. Given a corpus, this tool quickly and effectively expresses words in vector form through optimized training models, with core architectures including Continuous Bag-of-Words (CBOW) and Skip-gram. In principle, CBOW predicts target words from contextual semantics, while Skip-gram uses target words to predict sentence context. This study employs the Skip-gram model for word vector training because its performance is superior to CBOW on large datasets when appropriately configured and trained on sufficiently large corpora. The word vector training model structure is shown in Figure 1 [Figure 1: see original paper].

Assuming a vocabulary set $w_1, w_2, w_3, \ldots, w_n$, the Skip-gram model aims to maximize the following formula:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{i+j}|w_i)$$

where $c$ represents the number of context words before and after the current word. Larger $c$ values yield better training results but increase time consumption. In practical applications, appropriate $c$ selection combined with sufficiently large training corpora enables high-quality word vectors in relatively short time. In our experiments, text vocabulary is trained as 300-dimensional word vectors using the Skip-gram model on the Google News corpus containing 3 billion training instances. These word vectors are then constructed into text matrices for use as input data for CNN feature extraction.

## 2 CNN-ELM Hybrid Short Text Classification Model

Figure 2 [Figure 2: see original paper] illustrates the architecture of our proposed CNN-ELM hybrid model, which uses text matrices converted from word vectors as input. The convolutional feature extraction layer employs convolution kernels of different sizes to extract features from the input matrix, performs max pooling operations on the extracted vectors, concatenates them to obtain the

feature vector for the text matrix, and then uses multi-layer Highway networks for optimization. Finally, the optimized feature vector serves as input to the extreme learning machine classification layer to complete the final classification task.

Structurally, the convolutional feature extraction layer used in this model is a variant of the CNN structure proposed by Collobert et al. [21]. After processing through this layer, the text matrix is transformed into a feature vector.

### 2.1 Convolutional Feature Extraction Layer

**2.1.1 Word Vector Concatenation** For input data processing, if $v_i$ is the $k$-dimensional word vector at position $i$ in a sentence, $n$ is the length of the longest sentence in the corpus, and $l$ is the maximum width of convolution kernels used in the CNN, then the input data is a $k \times (n + l - 1)$ matrix representing a sentence. This matrix is constructed by concatenating the word vectors of all words in the sentence, which can be expressed as:

$$v_{1:n} = v_1 \oplus v_2 \oplus v_3 \oplus \cdots \oplus v_n$$

where $\oplus$ is the concatenation operator. In our model, for more convenient data processing, sentence length is set to a fixed value (the maximum sentence length in the dataset). Sentence matrices that do not reach this length are padded with $(l - 1)$ zero vectors, resulting in $n + l - 1$ word vectors, each with dimension $k = 300$.

**2.1.2 Convolution Operation** If a convolution kernel has width $h$ and dimension $k$, it represents a convolution window containing $h$ word vectors and uses these to generate a new feature. In our model, convolution kernel width $h$ is diversified, as combining features extracted from different convolution windows better reflects the true semantic features of a sentence. Assuming feature vector $a_i$ is generated from words $x_{i:i+h-1}$:

$$a_i = f(w \cdot x_{i:i+h-1} + b)$$

where $w$ is the convolution kernel weight, $b$ is the bias term, and $f$ is a non-linear activation function such as ReLU or Tanh. Convolution kernels similar to the above are applied to the sentence $\{x_{1:h}, x_{2:h}, \ldots, x_{n-h+1:n}\}$ to generate a feature vector:

$$a = [a_1, a_2, \ldots, a_{n-h+1}]$$

**2.1.3 Max Pooling and Feature Vector Output** After obtaining feature vectors generated by each convolution kernel, we apply max pooling to retrieve their maximum value $A = \max\{a\}$, then concatenate $A$ to obtain the feature vector extracted by the convolutional structure. Max pooling aims to capture the most representative features generated by different convolution kernels while effectively reducing spatial and temporal complexity.

**2.1.4 Dropout** To avoid overfitting during training, we employ Dropout to prevent some hidden neurons from participating in forward propagation, excluding them from the update process and making weight updates independent of fixed node interactions. The vector obtained after Dropout is then optimized using Highway networks.

### 2.2 Highway Network Optimization Layer

Inspired by LSTM, the Highway network proposed by Srivastava et al. is also a learnable gating mechanism that enables local adjustment of information flow for optimization. In a traditional feedforward neural network with $L$ layers, each layer applies a non-linear transformation $H$ with parameters $W_H$ to input $x_i$ to produce output $y_i$, expressed as $y = H(x, W_H)$. The Highway network adds two non-linear mapping functions $T$ and $C$, making the output $y$ become:

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C)$$

where $T$ is called the transform gate and $C$ is the carry gate. To simplify the model, the carry gate $C$ is typically set to $(1 - T)$, making the equation:

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$$

In this formulation, $x$, $y$, $H$, and $T$ must have the same dimensions; if dimensions are insufficient, zero-padding is applied. The Highway network processes input information flow through the transform gate, a highway-toll-like operation that modifies portions of the input information flow. This method has proven effective at addressing convergence difficulties in deep neural network training, thereby improving model performance.

In our model, the Highway network layer optimizes convolutional features to improve classification effectiveness. Additionally, due to the low complexity of Highway networks, using multiple layers to continuously optimize convolutional feature vectors does not excessively increase time and space overhead. We determine the optimal number of Highway network layers through experiments to obtain the best feature vectors.

### 2.3 Extreme Learning Machine Classification Layer

Since its proposal, Extreme Learning Machine has demonstrated excellent performance in classification and regression tasks. Given discretized input data and hidden node numbers, ELM can quickly compute results, making it widely applicable in image recognition and handwritten digit recognition. For $N$ distinct learning samples $(x_i, t_i)$ where $x_i \in \mathbb{R}^n$ and $t_i \in \mathbb{R}^m$, the basic principle of ELM can be expressed as:

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = o_j, \quad j = 1, 2, \ldots, N$$

where $g(\cdot)$ represents the hidden layer activation function, $\beta_i$ is the weight between the network output layer and the $i$-th hidden neuron, and $b$ is the bias of hidden layer neurons. This formula can be simplified to $H\beta = T$, where:

$$H = \begin{bmatrix} g(w_1, b_1, x_1) & \cdots & g(w_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ g(w_1, b_1, x_N) & \cdots & g(w_L, b_L, x_N) \end{bmatrix}_{N \times L}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$$

This network only requires setting the number of hidden layer nodes without adjusting input weights or hidden layer biases, and can calculate output weights between the hidden and output layers through the Moore-Penrose generalized inverse matrix $H^\dagger$ (where $H^\dagger$ is the pseudo-inverse of $H$) and the bias values $b_j$ of hidden layer neurons.

Although ELM offers fast computation and excellent generalization, obtaining its optimal solution typically requires manual statistical methods. The proposal of Error-Minimized Extreme Learning Machine (EM-ELM) [16] solved this problem by recursively adding hidden nodes to automatically calculate optimal results. Given training set $\{(x_i, t_i)\}_{i=1}^N$, maximum hidden nodes $L_{\max}$, and desired learning accuracy $e > 0$, EM-ELM calculation consists of two stages:

**Stage 1: Initialization Phase**
Initialize a single-hidden layer feedforward neural network (SLFN) and set initial hidden nodes $L_0$ (this paper uses $L_0 = 1$). Then calculate the hidden layer output matrix $H_1$ and output error $E(H_1) = \|H_1\beta_1 - T\|$ as in traditional ELM.

**Stage 2: Recursive Calculation Phase**
Let $k = 0$. While $L_k < L_{\max}$ and $E(H_k) > e$:
- $k = k + 1$, randomly add $\delta$ hidden nodes to the existing SLFN, making hidden nodes $L_{k+1} = L_k + \delta$ and hidden output matrix $H_{k+1} = [H_k, H_\delta]$.

- Update output weights using recursive methods: $\beta_{k+1} = \begin{bmatrix} \beta_k - D_k^T H_\delta^T \beta_k \\ D_k^T T \end{bmatrix}$, where $D_k = (I - H_k H_k^\dagger) H_\delta$.

End while.

During training, we first train the convolutional neural network layer and Highway network layer to convergence using the training set, then use the converged model to extract features from both training and test set text matrices. The extracted features serve as input to the error-minimized extreme learning machine, implementing the hybrid architecture shown in Figure 2, which yields optimal classification results after EM-ELM initialization and recursive calculation.

## 3 Experiments and Analysis

We evaluate the CNN-ELM hybrid short text classification model using multiple English short text classification datasets. These experiments were conducted on a server with 256GB memory and an Intel i7 4.0-GHz CPU, using Python 2.7 and TensorFlow.

### 3.1 Experimental Data

To assess our model, we use English short text datasets widely applied in recent text classification research to test classification accuracy and analyze performance. These datasets include MR[2], SST-1[3], Subj [22], TREC , Irony [23], Tweet , and Polite [24]. Additionally, we crawled user profile data from social media websites to construct the Description dataset with manual annotation, achieving a Kappa coefficient of 0.83 through consistency testing, satisfying dataset consistency requirements for account type classification tasks. Detailed information for each dataset is shown in Table 1 :

**Table 1 Dataset Details**

| Dataset | c | l | N | | V | |
|---|---|---|---|---|---|---|
| MR | 2 | 20 | 10662 | 18765 | 16448 | CV |
| SST-1 | 5 | 18 | 11855 | 17836 | 16262 | 2210 |
| Subj | 2 | 23 | 10000 | 23757 | 21913 | CV |
| TREC | 6 | 10 | 5952 | 9592 | 8684 | 500 |
| Irony | 2 | 19 | 28686 | 28007 | 22180 | CV |
| Tweet | 3 | 19 | 7088 | 23813 | 18544 | CV |
| Polite | 2 | 23 | 11472 | 19190 | 16805 | CV |
| Description | 4 | 15 | 10000 | 20421 | 17654 | 2000 |

Where $c$ represents the number of target categories, $l$ represents the average sentence length in the dataset, $N$ represents dataset size, $|V|$ represents dictionary size, $|Vpre|$ represents the number of vocabulary items present in the Google

pre-trained word vector dataset, and "CV" indicates that the dataset lacks a training/test split, requiring 10-fold cross-validation for performance testing.

### 3.2 Experimental Setup

**Word Vector Training:** We use the Google Word2Vec tool to train word vectors, with each word represented as a 300-dimensional vector using the Skip-gram model. For words not appearing in Word2Vec, we randomly initialize them with values in $[-0.5, 0.5]$.

**Convolutional Neural Network:**
- Input channels: CNN-non-static
- Convolution kernels: 100 kernels each of widths [3, 4, 5] with ReLU activation
- Mini-batch size: 50
- Dropout: 0.5 (training only)
- Optimizer: ADAM with learning rate 0.001, decaying 50% every 8 iterations

**Highway Network:**
- Layers: 2
- Gate bias: 0
- Non-linear transformation $H$: ReLU

**Error-Minimized Extreme Learning Machine:**
- Activation function: Sigmoid
- Maximum hidden nodes $L_{\max}$: Approximately equal to dataset size, as shown in Table 2
- Desired learning accuracy $e$: 90%
- Initial hidden nodes: Starting from 1, incrementing by 1 node each time

**Table 2 Dataset Capacity and Maximum Hidden Node Settings**

| Dataset | $L_{\max}$ |
|---|---|
| MR | 10662 |
| SST-1 | 11855 |
| Subj | 10000 |
| TREC | 5952 |
| Irony | 28686 |
| Tweet | 7088 |
| Polite | 11472 |
| Description | 10000 |

### 3.3 Experimental Results and Comparative Analysis

Our experiments first investigate the impact of Highway networks on classification results by testing different numbers of Highway network layers to determine the optimal configuration. We then compare our model with various machine learning algorithms, including traditional machine learning methods and their

variants (LDA, K-Nearest Neighbors, SVM, Naive Bayes, Random Forest, Decision Trees) and Mandelbaum et al.' s improved CNN text classification model. Finally, we examine how different network structures in our model affect classification results.

**3.3.1 Impact of Highway Networks on Classification Results** We first verify the impact of Highway networks by combining them with CNN classification models using different numbers of layers. Classification results on four different datasets are shown in Figure 3 [Figure 3: see original paper]. The results indicate that Highway networks optimize different datasets to varying degrees. On Irony and Polite datasets, Highway networks provide the most noticeable improvement, while on the Description dataset, they also yield slight enhancements. However, on the TREC dataset, adding Highway networks has almost no impact on results.

Regarding Highway network depth, results on Irony and Polite show that adding 2 Highway network layers yields optimal performance, while also improving upon the original model on the Description dataset. Therefore, we use 2 Highway network layers for feature optimization in subsequent models.

Theoretically, Highway networks were designed to address training difficulties in extremely deep neural networks. Highway networks with hundreds or even thousands of layers can be trained directly using gradient descent with various non-linear activation functions, and their optimization method is largely independent of network depth without excessive time and space overhead. However, in practice, limited training data causes excessive network layers to lead to overfitting and reduced generalization. Testing 100-layer Highway networks on the Polite dataset yielded results below 60%. Nevertheless, as data volumes increase in the future, extremely deep Highway networks still hold significant potential.

**3.3.2 Algorithm Comparison Table 3 Comparison with Different Machine Learning Algorithms (%)**

| Algorithm | Description |
|---|---|
| LDA | 85.6 |
| KNeighbors (k=4) | 88.3 |
| MultinomialNB | 89.2 |
| DecisionTree (entropy) | 88.5 |
| DecisionTree (gini) | 88.1 |
| RandomForest | 89.8 |
| Bagging | 90.1 |
| SVM (linear) | 90.5 |
| CNN-rand | 95.8 |
| CNN-static | 95.9 |
| CNN-non-static | 96.3 |
| **Our Model** | **97.3** |

| Algorithm | Description |
|---|---|

Table 3 shows experimental results on the Description dataset. The emergence of convolutional neural networks has broken previous records held by traditional machine learning algorithms. CNN-rand and CNN-non-static achieve similar results because network data contains numerous new words, emoticons, and links, causing nearly half of the vocabulary in this dataset to be absent from Word2Vec. This makes randomly initialized word vectors constitute a large proportion of the dictionary, making CNN-rand performance comparable to CNN-non-static. Building upon CNN-non-static, our model first uses Highway networks to optimize CNN-extracted features, improving classification accuracy by 1% over CNN-non-static. Combining this with extreme learning machine classification principles further raises classification results to 97.3%.

**Table 4 Results Comparison Across Different Datasets (%)**

| Dataset | CNN-non-static | Our Model |
|---|---|---|
| MR | 81.3 | 82.1 |
| SST-1 | 48.2 | 49.5 |
| Subj | 93.2 | 93.6 |
| TREC | 93.2 | 94.0 |
| Irony | 73.2 | 75.0 |
| Tweet | 66.2 | 66.8 |
| Polite | 68.5 | 70.1 |
| Description | 96.3 | 97.3 |

We further test our model' s generalization capability using multiple short text classification datasets, with comparison results against the improved CNN-non-static model shown in Table 4. Our model achieves higher classification accuracy on most datasets. On the TREC dataset, Figure 3 already indicated that Highway networks cannot effectively improve classification, but our model ultimately improves results by 0.8%, further demonstrating the effectiveness of combining extreme learning machine classification principles. Meanwhile, since 2-layer Highway networks provide significant optimization on Irony and Polite datasets, achieving 75% and 70.1% accuracy respectively even without ELM combination, integrating with extreme learning machine further enhances model classification performance.

### 3.3.3 Impact of Model Structure on Classification Results  Table 5 Results of Different Model Structures (%)

| Model | Description | SST-1 | Subj | Tweet | Polite |
|---|---|---|---|---|---|
| CNN-non-static | 96.3 | 48.2 | 93.2 | 66.2 | 68.5 |
| CNN-ELM | 96.8 | 48.6 | 93.4 | 66.5 | 69.2 |
| CNN-1Highway | 97.0 | 48.8 | 93.5 | 67.1 | 69.8 |
| CNN-2Highway | 97.1 | 49.0 | 93.4 | 66.8 | 70.1 |
| CNN-1Highway-ELM | 97.2 | 49.2 | 93.6 | 66.9 | 69.9 |
| CNN-2Highway-ELM | **97.3** | **49.5** | **93.6** | 66.8 | **70.1** |

Our model combines CNN with Highway networks and extreme learning machine principles for classification. Table 5 compares different combinations of these three structures across datasets. In most cases, combining only Highway networks provides greater improvement than combining only extreme learning machines. On the Tweet dataset, the model with only 1 Highway network layer even achieves the highest accuracy. Generally, combining extreme learning machine after Highway network optimization yields the best results. Meanwhile, experiments on the Description dataset show that selecting the correct number of Highway network layers also significantly impacts optimal results. In summary, obtaining the optimal model for specific tasks still requires multiple parameter and model adjustments. However, manual testing to determine the best model is overly complex. If Highway networks could automatically select the optimal number of layers for classification tasks, the reliability of this hybrid model would substantially improve.

## 4 Conclusion

This paper proposes a CNN-ELM hybrid short text classification model that combines extreme learning machine and Highway network theories with convolutional neural networks, achieving superior classification results compared to original models. Experiments demonstrate that this approach is more effective than traditional machine learning algorithms and convolutional neural network models. Future research will focus on: (1) Improving Highway networks by adding automatic selection of optimal network layers to enhance hybrid model reliability; (2) Adding external features combined with convolutional features to improve classification performance; (3) Improving CNN feature extraction structures by combining with other deep neural networks to form deeper architectures that extract more representative features.

## References

[1] Cover T M, Thomas J A. Elements of information theory [M]. 2nd ed. New Jersey: Brooks//John Wiley & Sons, 2012.

[2] Cai Lijuan, Hofmann T. Text categorization by boosting automatically extracted concepts [C]// Proc of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2003: 75-82.

[3] Hingmire S, Chougule S, Palshikar G K, et al. Document classification by topic labeling [C]// Proc of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2013: 877-880.

[4] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model [J]. Journal of machine learning research, 2003, 3 (Feb): 1137-1155.

[5] Mikolov T, Sutskever I, Chen Kai, et al. Distributed representations of words and phrases and their compositionality [C]// Neural Information Processing Systems. 2013: 3111-3119.

[6] Kim Y. Convolutional neural networks for sentence classification [C]// Proc of Conference on Empirical Methods in Natural Language Processing. 2014: 1746-1751.

[7] Mandelbaum A, Shalev A. Word embeddings and their use in sentence classification tasks [J/OL]. arXiv preprint, 2016, arXiv: 1610. 08229 [2017-11-30]. https://arxiv.org/abs/1610.08229.

[8] Chen Zhao, Xu Ruifeng, Gui Lin, et al. Chinese sentiment analysis combining convolutional neural networks and word sentiment sequence features [J]. Journal of Chinese Information Processing, 2015, 29 (6): 172-178.

[9] Liu Longfei, Yang Liang, Zhang Shaowu, et al. Microblog sentiment orientation analysis based on convolutional neural networks [J]. Journal of Chinese Information Processing, 2015, 29 (6): 159-165.

[10] Tong Bohui, Fu Kun, Huang Yu, et al. Entity relation extraction based on multi-channel convolutional neural networks [J]. Application Research of Computers, 2017, 34 (3): 689-692.

[11] Ebert S, Vu N T, Schütze H. CIS-positive: combining convolutional neural networks and svms for sentiment analysis in Twitter [C]// Proc of the 9th International Workshop on Semantic Evaluation. 2015: 527-532.

[12] Matsugu M, Mori K, Suzuki T. Face recognition using SVM combined with CNN for face detection [C]// Proc of International Conference on Neural Information Processing. 2004: 356-361.

[13] Huang Guangbin, Zhou Hongming, Ding Xiaojian, et al. Extreme learning machine for regression and multiclass classification [J]. IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics, 2012, 42 (2): 513-529.

[14] Huang Guangbin, Zhu Qinyu, Siew C K. Extreme learning machine: a new learning scheme of feedforward neural networks [C]// Proc of IEEE International Joint Conference on Neural Networks. 2004: 985-990.

[15] Yu Jiasheng, Chen Jin, Xiang Z Q, et al. A hybrid convolutional neural networks with extreme learning machine for WCE image classification [C]// Proc of IEEE Conference on Robotics and Biomimetics. 2015: 1822-1827.

[16] Feng Guorui, Huang Guangbin, Lin Qingping, et al. Error minimized extreme learning machine with growth of hidden nodes and incremental learning [J]. IEEE Trans on Neural Networks, 2009, 20 (8): 1352-1357.

[17] Srivastava R K, Greff K, Schmidhuber J. Highway networks [J/OL]. arXiv preprint, 2015, arXiv: 1505.00387 [2017-11-30]. https://arxiv.org/abs/1505.00387.

[18] Srivastava R K, Greff K, Schmidhuber J. Training very deep networks [C]// Advances in Neural Information Processing Systems. 2015: 2377-2385.

[19] Kim Y, Jernite Y, Sontag D, et al. Character-Aware Neural Language Models [C]// Proc of the 30th AAAI Conference on Artificial Intelligence. 2016: 2451-2457.

[20] Hsu W N, Zhang Y, Lee A, et al. Exploiting depth and highway connections in convolutional recurrent deep neural networks for speech recognition [C]// Proc of InterSPEECH. 2016: 395-399.

[21] Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch [J]. Journal of Machine Learning Research, 2011, 12 (Aug): 2493-2537.

[22] Pang Bo, Lee L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts [C]// Proc of the 42nd Annual Meeting on Association for Computational Linguistics. 2004: 271.

[23] Wallace B C, Do Kook Choe L K, et al. Humans require context to infer ironic intent (so computers probably do, too) [C]// Proc of the 52nd Annual Meeting of the Association for Computational Linguistics. 2014: 512-516.

[24] Danescu-Niculescu-Mizil C, Sudhof M, Jurafsky D, et al. A computational approach to politeness with application to social factors [J/OL]. arXiv preprint, 2013, arXiv: 1306.6078 [2017-11-30]. https://arxiv.org/abs/1306.6078.

*Note: Figure translations are in progress. See original paper for figures.*