Hindawi
Complexity
Volume 2020, Article ID 8516216, 11 pages
https://doi.org/10.1155/2020/8516216

Research Article

Joint Character-Level Convolutional and Generative Adversarial Networks for Text Classification

Tianshi Wang,[1] **Li Liu**,[1,2] **Huaxiang Zhang**,[1,2] **Long Zhang**,[1] **and Xiuxiu Chen**[1]

[1]*School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China*
[2]*Institute of Data Science and Technology, Shandong Normal University, Jinan 250014, Shandong, China*

Correspondence should be addressed to Li Liu; liuli_790209@163.com

Received 23 March 2020; Accepted 15 April 2020; Published 30 April 2020

Academic Editor: Jianquan Lu

With the continuous renewal of text classification rules, text classifiers need more powerful generalization ability to process the datasets with new text categories or small training samples. In this paper, we propose a text classification framework under insufficient training sample conditions. In the framework, we first quantify the texts by a character-level convolutional neural network and input the textual features into an adversarial network and a classifier, respectively. Then, we use the real textual features to train a generator and a discriminator so as to make the distribution of generated data consistent with that of real data. Finally, the classifier is cooperatively trained by real data and generated data. Extensive experimental validation on four public datasets demonstrates that our method significantly performs better than the comparative methods.

1. Introduction

Machine-learning models have achieved remarkable results in computer vision (CV), automatic speech recognition, and neural language processing (NLP). With the development of artificial neural networks, text classification becomes one of the most intriguing fields of NLP [1, 2]. Text classification refers to the division of texts in the corpus into predefined categories based on its contents and other attributes. It is widely used in many applications [3–9], such as spam filtering, news categorization, sentiment analysis, and digital library.

As typical supervised learning (SL), text classification requires abundant manually labeled samples for training. However, manually adding ground truth to vast amounts of texts is difficult to achieve in practical application, so the number of labeled samples is not enough to meet the requirements. For small training samples, the insufficient depiction leads to the poor generalization ability of classifiers obtained from learning.

To improve the generalization ability of classifiers, the feasible solution is to improve learning algorithms or increase training samples. The method worth mentioning is support vector machines (SVM), which improves the generalization ability on unknown samples by learning the hyperplane of the maximum interval between different categories. SVM alleviates the problem of small samples to some extent, but the high time complexity leads to the limitation of engineering applications, especially online information processing. Besides, the existence of massive unlabeled samples has gradually drawn the attention of scholars to semisupervised learning (SSL). Most of the semisupervised classification methods train the objective classifier through an initial classifier until it reaches the convergence condition. However, such methods have high-computational complexity and may bring about large-scale sample problems.

Compared with the computationally expensive SSL, generative adversarial nets (GAN) proposed by Goodfellow et al. [10] provides an effective method for generating new data. In this paper, we employ GAN to generate the textual samples according to the data distribution of the input samples and label the generated samples. Furthermore, we use Char-level CNN to extract the text semantics and utilize the textual generative network to increase the training samples. To summarize, our contributions are listed as follows:

(1) The novel structure of Char-level CNN is designed for small-scale datasets to generate textual features. By optimizing the configuration of the convolutional and pooling layers, the output comprehensively inherits the text semantics. Besides, the dense network is designed deeper because the generated data reduces the risk of its overfitting.

(2) The data augmentation module based on high-level semantic is proposed. The text semantics are quantified and directly input into the network as real data to obtain various textual features. The module not only avoids the feature extraction of the generated texts and saves the computing resources but also describes the overall distribution characteristics of data to make the classifier perform better on small-scale datasets.

The rest of this paper is structured as follows. Section 2 summarizes the related work of textual feature construction, text generation, and semisupervised learning. Section 3 details our model to solve the small-scale datasets text classification problem. Section 4 presents the experimental setting and results analysis. Section 5 concludes our whole work and gives further research direction.

2. Related Work

Compared with other media streams, the most special aspect of texts is that semantic information is difficult to express. For subtasks in NLP, how to quantify abstract semantic information is particularly critical. Therefore, the final performance of text classification is jointly affected by the classification models and feature representation methods [11–13].

The mainstream representation methods for text classification can be roughly divided into three categories. Traditional textual features are mainly generated through methods such as bag of words (BOW) and n-gram. Both are usually combined with term frequency-inverse document frequency (TF-IDF) and other element features as textual features. However, the momentous drawback of these methods is that they ignore the context and order of words. The second language model is based on attention mechanisms, commonly known as hierarchical attention and self-attention, which extracts textual features by scoring input words or sentences differentially. In the third language model, text can be represented by sequence or structured models through the introduction of artificial neural networks. In addition to the above three language models, some pretraining models, such as XLNet and BERT, have also been proposed for NLP. However, these pretrained models require large amounts of labeled textual data and are not suitable for new categories and small samples of textual data.

In the text representation methods based on the sequence or structured models, Bengio et al. [14] first tried to use neural networks to produce dense, low-dimensional vectors for words. Mikolov [15] proposed a language model, called Word2Vec, which can transform each word into vector form according to the context. The model can take the representative words as the representation of text by working with clustering algorithms. Besides, textual features can be obtained by simply combining the word embeddings to replace sentences or texts, and the linear model FastText [16] is widely used in text classification in this way. Kim [17] applied the convolutional neural network to the classification process and obtained the textual features by processing the matrix formed by word embedding. Kaichbrenner et al. [18] proposed a convolutional architecture and dubbed the dynamic convolutional neural network (DCNN) for sentence modeling. Zhang et al. [19] proposed the use of convolutional networks for text classification at the character level, but their network structures only work well on large-scale datasets. Then, for the structural configuration of convolutional neural networks, Le et al. [20] studied the importance of depth in convolutional models. At the same time, the recurrent neural networks are also applied to the language models due to their memorability and Turing completeness. Chung et al. [21] compared different types of recurrent units, especially gating mechanisms such as long short-term memory (LSTM) and gated recurrent unit (GRU). Zhu et al. [22] attempted to build structured representations using prespecified parsing trees. Recently, Zhang et al. [23] proposed a reinforcement learning (RL) method to get structured sentence vectors. Note that before the abovementioned methods were proposed, some traditional machine-learning algorithms were widely used in text classification. For example, k-nearest neighbor (KNN) for classification by measuring the distance between different features, decision tree combining information entropy and tree structure, and Naive Bayes based on Bayesian theory and characteristic conditional independence hypothesis. However, it is proved that the performance of machine-learning algorithms is lower than that of the methods based on deep learning in the text classification task.

To solve the problem of insufficient training samples, many methods of data augmentation and semisupervised learning can be used to improve the performance of classifiers. Wei and Zou [24] presented an easy data augmentation (EDA) technique for boosting performance on text classification tasks. Although EDA reduces overfitting when training on smaller datasets, the improvement is at times marginal. Wang and Wu [25] proposed a framework that combines variational autoencoder (VAE) and neural networks to deal with text classification and generation tasks. GAN [10, 26] was firstly proposed for continuous data (image generation, inpainting, style transfer, etc.) and has shown excellent performance in computer vision (CV). Yu et al. [27] extended GAN to discrete and sequential data to alleviate the above deficiency. Since then, various text generation methods have been proposed via GAN. Xu et al. [28] proposed a text generation model called DP-GAN, which can encourage the generator to produce diverse and informative text. Li et al. [29] combined reinforcement learning, GAN, and recurrent neural networks to build a category sentence generative adversarial network. Miyato et al. [30] extended adversarial and virtual adversarial training to the text domain. Ahamad [31] also tried to solve the above problem by using Skip-Thought sentence

embeddings in conjunction with GANs. Although the methods utilize text generation and feature reconstruction to alleviate the problem of small-scale datasets, the classification performance is still difficult to further improve due to the large work of feature engineering and the trouble of textual feature extraction.

3. Methodology

In the section, the character-level convolutional and generative adversarial networks (CCNN-GAN) are utilized to set up a novel hybrid text classification framework in Figure 1. In contrast to the implementation of continuous data, the textual data is modelled by convolutional networks and character quantization in our model. Char-level CNN embeds the texts in the corpus into fixed-length features, and then the features are input into the generative adversarial network (GAN) and backpropagation network (BP network), respectively. The generated data not only enriches the textual features but also effectively solves the problems of insufficient samples and single information when dealing with small-scale datasets. After that the real samples and generated samples from GAN are mixed into a BP network for training. The design is modular, and the text information is transmitted between modules in the form of processed features.

3.1. Text Quantization Module. The acceptable encoding of Char-level CNN includes alphabetic encoding, utf-8 encoding, and pretrained character embedding vector. Since the proposed model is mainly used in the English-dominated alphabetic attachment language, alphabetic encoding is applied to the text quantization process. In the embedding layer, features of encoded characters are used as input. An alphabet of size α is stipulated; then, an embedded dictionary is created and an embedding matrix is formed based on the alphabet. Null character and characters that do not exist in the alphabet are replaced by an all-zero vector. The quantization length of the character feature is set to β. Assume [19] that β characters in the text can reflect the content of the text and the part that exceeds length β is ignored.

The foundational alphabet ($\alpha = 45$) and elaborate alphabet ($\alpha = 70 + \alpha_0$) are stipulated, where α_0 is the length of the auxiliary characters, and the details are shown in Table 1. The foundational alphabet used in the proposed model consists of 45 characters which are 36 English letters and Arabic numerals, 8 other characters, and the null character. The alphabet is applicable to most documents. For the corpus with high symbol content, the foundational alphabet is supplemented by the elaborate alphabet. The elaborate alphabet consists of 25 symbol characters and auxiliary characters of varying lengths. Char-level CNN consists of 8 convolution layers, 3 pooling layers, and 3/4/5 fully connected layers. The configuration of the convolutional neural network is shown in Table 2 and Figure 2.

The classifier that is composed of fully connected layers is discussed in detail in Section 3.3. The Char-level CNN is constructed to compute 1D convolution and the weights are initialized using Gaussian distribution. The mean and standard deviation to initialize the model is (0, 0.05). The pooling layers are also applied between the convolutional layers for increasing the area covered with the next receptive fields.

In addition, a rectified linear unit (ReLU) is taken as the activation function in the classifier, and local response normalization (LRN) [32] is added behind each pooling layer. The LRN imitates the biological neural system layer of lateral inhibition mechanism and improves the generalization ability of the model. The function is as follows:

$$L^i = \frac{P^i}{\left(k + a \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (P^j)^2\right)^b}, \quad (1)$$

where P is the tensor obtained after pooling, i and j represent the ith and jth kernel, k, n, a, and b are hyperparameters, and N is the total number of kernels.

3.2. Data Augmentation Module. Different from the typical RL setting, the data augmentation module enriches the predetermined corpus at the semantic level. Specifically, the module is an adversarial network, in which the generator can generate directly many textual features with the same distribution of the processed real-world texts and the discriminator takes the convolutional textual data as real data. Thus, given the processed textual features as input, the adversarial network can generate various generated features that contain diverse and informative text semantics.

To optimize the performance of the overall framework, the data augmentation module is simplified. The output of the module is not in the form of sentences or documents but in the form of textual features that contain semantic information. Based on the abovementioned ideas, the adversarial network no longer needs to connect with the structure that converts textual features into sentences or documents. During the network training, the processed features and its category are all input into the adversarial network so that it can output textual features that belong to each category.

More formally, the input data $X_{1:m} = \{X_1, X_2, X_3, \ldots, X_i, \ldots, X_m\}$ of m categories comes from the processed corpus Γ, and the high-level textual features is denoted as $X_i = x_{i,1:n} = \{x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,i}, \ldots, x_{i,n}\}$, where $x_{i,j}$ refers to the jth textual feature of category i. The generative network outputs multiple categories of labeled textual features $X^*_{1:m} = \{X^*_1, X^*_2, X^*_3, \ldots, X^*_i, \ldots, X^*_m\}$, where $X^*_i = x^*_{i,l:n} = \{x^*_{i,1}, x^*_{i,2}, x^*_{i,3}, \ldots\}$ refers to the generated features of category i.

The representation of corpus in Section 3.1 show that the data points $\{x_{i,j}\}$ are independent of each other and identically are taken from real-world distribution $p_{\text{data}}(x)$. The generative network is to learn a generator's distribution $p_g(x)$ that gradually approximates to $p_{\text{data}}(x)$. The value function of conditional GAN is defined as follows:

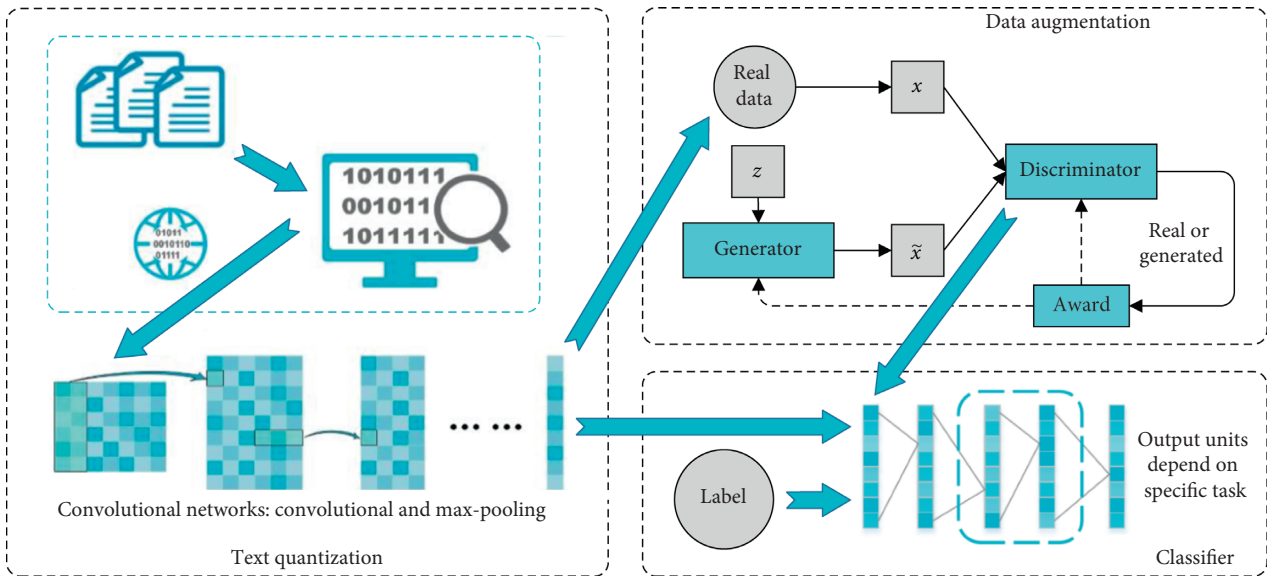

Figure 1: The framework of text classification.

Table 1: The nonspace characters of alphabets.

Alphabets	Nonspace characters
Foundational	a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9, ; . ! ? : ( )
Elaborate	- ’ “ / \ | _ @ # $ % ^ & ? ~ 0 + − = < > [ ] { } … …

Table 2: The parameter setting of the convolutional neural network.

Layer	Feature	Kernel	Pooling	Stride	
				Conv	Pool
1	256	5	N/A	1	3
2	256	5	3	1	3
3	256	5	N/A	1	3
4	256	5	3	1	3
5	256	3	N/A	1	3
6	256	3	N/A	1	3
7	256	3	N/A	1	3
8	256	3	3	1	3

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x \mid y)] + \mathbb{E}_{z\sim p_z(Z)}[\log(1 - D(G(z \mid y)))], \tag{2}$$

where x is the real textual feature, y is the category label corresponding to feature x, and z is random noise. Through the different settings of y, the textual features of different categories can be obtained in the process of text generation. Note that GAN is designed for continuous data, that is, it can only be constructed by differentiable functions. In the correlation processing of discrete data, it is difficult to transfer the gradient of the discrete outputs to the discriminator, so the discriminator cannot be updated. There are many feasible methods to solve the problem. This paper uses the policy gradient [27] to improve the application field of the adversarial network. The generator and discriminator are trained alternatively and the detail of the adversarial model is as follows.

3.2.1. The Generator for Textual Data. Recurrent neural network (RNN) is designed to solve the vanishing and exploding gradient in backpropagation, and it is widely used in NLP because of the discrete distribution of textual data. The GRU is set as the generative network and the structure is shown in Figure 3. The update gate z_t is used to control how much the previous state information is brought into the current state. The higher the value of the z_t is, the more state information is brought at the previous moment. The function of the updated gate is as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \tag{3}$$

where $[h, x]$ is the vector concatenation. The reset gate r_t controls how much information is written to the current candidate set $\tilde{h}_t$ from the previous state. The smaller the r_t, the less information is written from the previous state. The function of the reset gate is as follows:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]). \tag{4}$$

Since each unit has its reset and update gates, each hidden unit learns dependencies on different scales. The units that learn to capture short-term dependencies activate

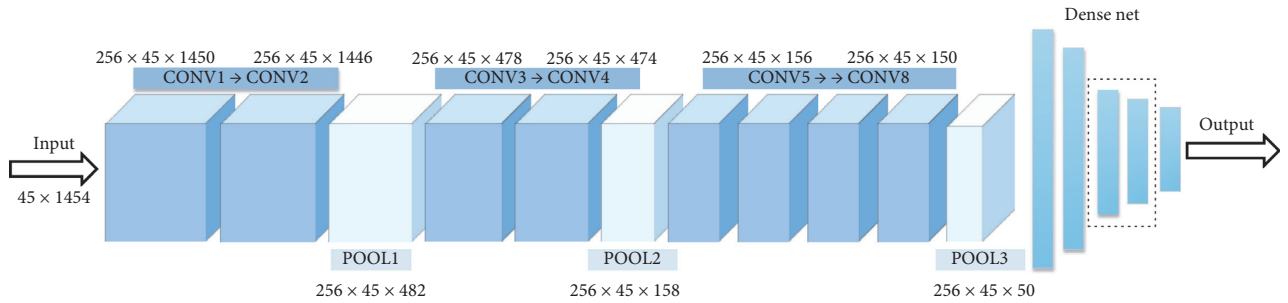

Figure 2: The structure diagram of the convolutional neural network.

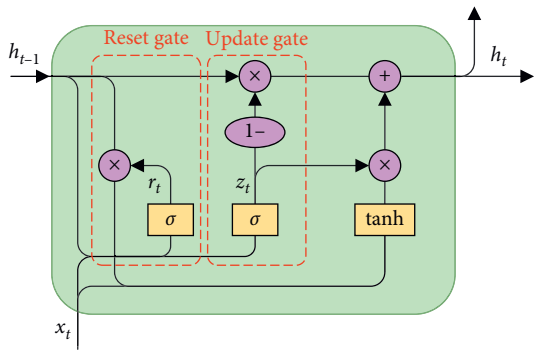

Figure 3: The detailed structure of the gated recurrent unit. z_t and r_t are update gate and reset gate, respectively.

the reset gates, while the units that capture long-term dependencies activate the update gates. The update functions are as follows:

$$\begin{aligned} \widetilde{h}_t &= \tanh\left(W_{\widetilde{h}} \cdot \left[r_t \odot h_{t-1}, x_t\right]\right), \\ h_t &= \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \widetilde{h}_t, \end{aligned} \tag{5}$$

where $\odot$ is the elementwise product. We use random noise as input into the generator to construct the mapping of noise space to text semantic space.

3.2.2. The Discriminator for Data Screening. Since both the real text and the generated text can be quantified as a feature of fixed length, a convolutional network is constructed to discriminate the source of the text. The textual feature $\{x_1, x_2, \ldots, x_n\}$ is processed as follows:

$$s_i = f\left(x_i \otimes w + b\right), \tag{6}$$

where $f(\cdot)$ is a nonlinear function, x_i is the l-dim textual features, $\otimes$ is the convolution operation, $w \in \mathbb{R}^k$ is a 1D kernel to produce a new feature map, and b is a bias term. The various numbers of kernels with different window sizes are used to extract different features. Specifically, the textual feature extracted by the kernel w with window size k is represented as follows:

$$s_i = \left[s_{i,1}, s_{i,2}, \ldots, s_{i,l-k+1}\right]. \tag{7}$$

Finally, the max-pooling operation is performed on the feature map $\widetilde{s} = \max\{s\}$ and all pooling features from different kernels are transferred to a fully connected softmax layer to get the probability that a given feature is real. When optimizing discriminative models, supervised training is applied to minimize the crossentropy, and the objective function is as follows:

$$\mathcal{H}(x, q) = -p(x)\log(q(x)) - (1 - p(x))\log(1 - q(x)), \tag{8}$$

where $p(x)$ is the real label of the textual features and $q(x)$ is the predicted probability from the discriminator.

3.3. Classifier Module. The classifier constructed by dense networks is a multilayer-feedforward network based on error backward propagation algorithm. The principle is to calculate the difference between the actual output and the expected output recursively, and the network adjusts the weights according to the difference.

The real features $X_{1:m}$ and generated features $X^*_{1:m}$ are input to the network for training. Here, ReLU is used as the activation. To explore the optimal network structure, the dense network consisting of three to five fully connected layers is constructed, as shown in Figure 4. Besides, the dropout modules are inserted between the fully connected layers to prevent overfitting, and the dropout probability is 0.5. The real semantics $X_{1:m}$ and generated semantics $X^*_{1:m}$ are input to the network for training. The activation function and its derivative are as follows:

$$\begin{aligned} f(x) &= \max(0, x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases} \\ f'(x) &= \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases} \end{aligned} \tag{9}$$

4. Experiments

In this section, we detail the experimental content and related setting, involving the datasets, the baselines, parameter setting, and experimental result analysis. To effectively demonstrate the advantages of the proposed method, we not only compare different text classification methods but also make the comparison of influence factors of our method. In this way, we further study the feasibility of the method through comparative experiments.

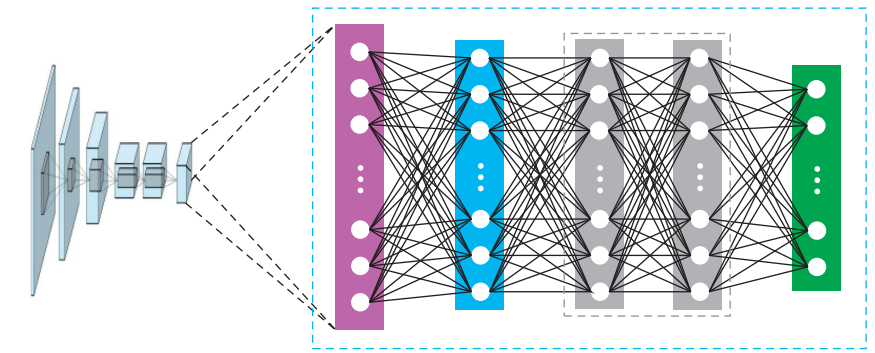

Figure 4: The diagram of the classifier structure.

4.1. Datasets. We use four corpora to evaluate the proposed framework. These datasets are

AG-News: original AG-News has over one million news articles collected from over 2,000 different news sources. We extract the data from the four largest categories in the original dataset, including world, environment, sport, and business news, and each category contains 30,000 for training and 1,900 for testing. The dataset can be downloaded from http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

DBPedia [33]: DBPedia ontology dataset is composed of 14 nonoverlapping categories from Wikipedia. In the experiment, we adopt the updated corpus by Zhang et al. Each category contains 40,000 training samples and 5,000 testing samples.

20NG: 20 newsgroup dataset is one of the international standard datasets for text classification, text mining, and information retrieval research. The dataset collects about 20,000 newsgroup documents that are evenly divided into newsgroup collections of 20 topics. Some newsgroups are dedicated to similar subjects, and some are completely unrelated. The dataset can be downloaded from http://qwone.com/~jason/20Newsgroups.

IMDB: the dataset is widely used for binary sentiment classification of movie reviews. It provides 25,000 highly polar movie reviews for training, 25,000 for testing, and additional unlabeled data. The dataset comes from http://ai.stanford.edu/~amaas/data/sentiment.

Note that most open datasets for text classification consumed huge resources to artificially tag it. To simulate the real-world small-scale datasets, we randomly extract parts of these datasets in the experiment. The details of the datasets are shown in Table 3.

4.2. Baselines. To make the experimental comparison more comprehensive and objective, we reproduce the mainstream text classification method, such as FastText, DPCNN, LEAM, and Virtual Adversarial. The details of these methods are as follows:

Table 3: Dataset statistics.

Dataset	Before extracting			L	K	α
	Train	Test	Unlabeled			
AG-News	120 k	7.6 k	—	46	4	5
DBPedia	560 k	70 k	—	56	14	5
20NG	11314	7532	—	221	20	10
IMDB	25 k	25 k	50 k	239	2	10

K is the number of categories, L is the average length of a document, and α is the extracting times of the sample sets.

Tree-LSTM [34]: the model proposed by Tai et al. extends the LSTM of sequence to the tree structure, that is, it can skip (or ignore) the whole subtree that has little effect on the result through the forgetting gate mechanism of LSTM, rather than just some subsequences that may have no linguistic significance.

Self-Attentive [35]: a model for extracting an interpretable sentence embedding by introducing self-attention. The method uses a 2D matrix to represent the embedding and proposes a self-attention mechanism and a special regularization term for the model.

Emb-CNN [17]: the model is a slight variant of the CNN architecture of Collobert et al. It shows that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. Kim additionally proposes a simple modification to the architecture to allow for the use of both task-specific and static vectors.

Char-CNN [19]: the method treats text as a kind of raw signal at character level and applies temporal ConvNets to it. The most important conclusion from the method is that character-level ConvNets can work for text classification without the need for word embedding.

Char-CRNN [36]: a architecture that utilizes both convolution and recurrent layers to efficiently encode character inputs. Compared with character-level convolution-only models, it can achieve comparable performances with much fewer parameters.

FastText [16]: the linear models with a rank constraint and fast loss approximation are often on par with deep-learning classifiers in terms of accuracy, and many orders of magnitude can be improved in evaluation.

L-MIXED [37]: a training strategy, even a simple BiLSTM model with crossentropy loss, can achieve competitive results compared with more complex methods. In addition to crossentropy loss, by using a combination of entropy minimization, adversarial, and virtual adversarial losses for both labeled and unlabeled data, the method can also perform very well.

DPCNN [38]: a low-complexity word-level deep convolutional neural network architecture for text classification that can efficiently represent long-range associations in text. Johnson et al. studied deepening of word-level CNNs to capture global representations of text and found a simple network architecture with which the best accuracy can be obtained by increasing the network depth without increasing computational cost by much.

LEAM [39]: the method of considering text classification as a label-word joint embedding in which each label is embedded in the same space with the word vectors. It maintains the interpretability of word embedding and has a built-in ability to leverage alternative sources of information, in addition to input text sequences.

Ad-Training [30]: the framework extends adversarial and virtual adversarial training to the text domain. The method applies perturbations to the word embedding in recurrent neural networks rather than to the original input itself.

Text GCN [40]: the model is initialized with one-hot representation for word and document, and it then jointly learns the embeddings for both words and documents, as supervised by the known class labels for documents.

4.3. Implementation Details. The 5/10-fold crossvalidation is applied to each dataset. The reduced training sets are marked as a part-dataset part-i ($i = 1, 2, 3, \cdots, 8$), where the amount of text in the dataset is, respectively, 200, 400, 800, 1,500, 2,500, 4,000, 6,000, and 10,000. To reduce the occasionality brought by sample selection, we use the same test set for multisize training sets. The total test sets are used to evaluate the performance on the 20NG dataset, and the other datasets are tested with 10,000 samples, respectively. Note that, on the IMDB, entire unlabeled samples are always provided for the semisupervised methods.

The baseline used in this paper replicates and sets parameters basically according to the original literature. The special cases are as follows: L-MIXED has two objective functions, the crossentropy loss $\mathscr{L}_{\mathrm{ML}}$ is adopted on the supervised datasets, and the mix function $\mathscr{L}_{\mathrm{MIXED}}$ is adopted on the IMDB. The unsupervised embeddings obtained by *tv*-embedding training in DPCNN. Adversarial Training has several training strategies, the virtual adversarial method based on unidirectional LSTM is utilized on the supervised datasets, and the bidirectional LSTM with virtual adversarial training on the IMDB.

In the details of our method, the elaborate alphabet length is set to 25, the convolution operation is set to "valid" and the pooling operation is the max-pooling, $\beta = 1454$. The number of output units for the last layer is determined by the problem in the classifier, that is, for the DBPedia it is 14. Besides, the other fully connected layers, all have 4096 units. During the training process, CCNN-GAN without the data augmentation module is first trained. Then, the parameters of the Char-level CNN are fixed and the data augmentation module is introduced to conduct incremental training of the classifier.

4.4. Results and Analysis. We analyze in detail the effect of different model settings, including the size of the alphabet, the number of generated features, and the structure of the classifier network. Besides, we compare the performance of the proposed method with those of the representative methods on the benchmark datasets.

4.4.1. Model Parameter Analysis. We use AG-News to analyze the impact of different settings. The alphabet size directly affects the time and efficiency of the classification method. We employ the foundational alphabet and the elaborate alphabet with the auxiliary symbols, respectively. As shown in Figure 5, the classification accuracy of the proposed method significantly improves with the growth of the dataset. When the scale increases to Part-6, the growth of the classification accuracy tends to be flat, which indicates that the dataset has a similar representational ability to the complete dataset.

However, the classification accuracy of the synthetic alphabet (the foundational alphabet and the elaborate alphabet) is similar to that of the foundational alphabet. Although the elaborate alphabet increases the representativeness of textual features, the improvement is negligible. Besides, with the alphabet increase, the network training time significantly increases.

To get the optimal network settings, we adjust the classifier structure and the amounts of samples by the generative network, respectively. Firstly, the number of fully connected layers and neurons in each layer is adjusted continuously to find the optimal configuration. Then, we adjust the quantity of generated texts and analyze the impact of generated samples on the accuracy. We compare the effects of the abovementioned variables on AG-News. As shown in Figure 6, the increase of the fully connected layers improves the accuracy of the classifier, but its change is not obvious.

When the number of real-world samples is small, the proposed framework can improve the performance of the

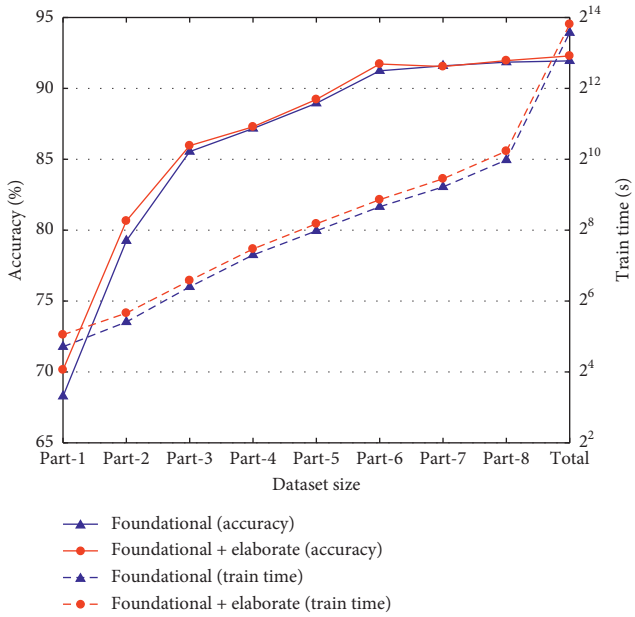

Figure 5: The impact of the alphabet size in CCNN-GAN.

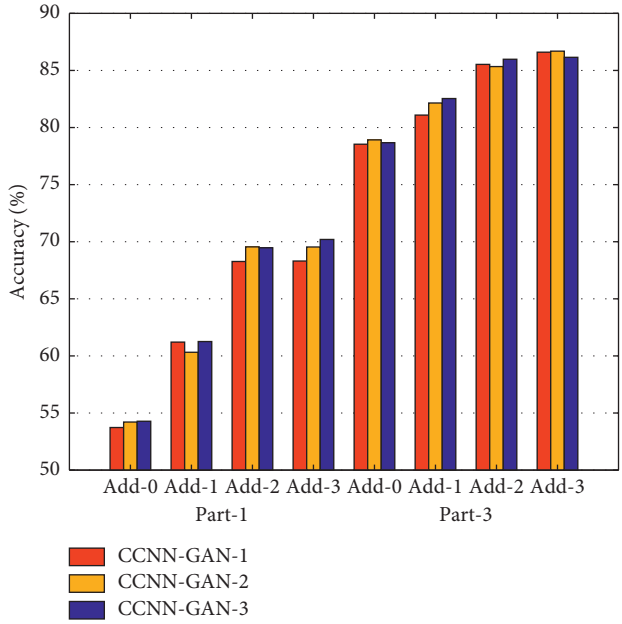

Figure 6: The impact of the number of generated features and the structure of the classifier network. Add-i ($i = 0, 1, 2, 3$) represents the number of samples generated by GAN, and the quantity of generated samples is 0, 0.5, 1, and 2 times of the real text quantity, respectively. CCNN-GAN-j ($j = 1, 2, 3$) represents the layers of dense networks, and the layers are 3, 4, 5, respectively.

classifier. However, when the number of generated samples continues to increase after reaching a certain level, the classifier is not significantly improved, which may be because the scale of original data limits the performance of the generative network, further limits the generation space of text, and finally affects the abstract semantic space of samples.

4.4.2. Comparison of Different Methods. We test different methods on the datasets mentioned in Section 4.1. To improve the efficiency on the premise of better accuracy, the structure setting of CCNN-GAN is as follows: the alphabet is foundational, the number of fully connected layers is three, and the number of generated textual features is the same as the number of real samples.

CCNN-GAN is superior to all the comparison methods when dealing with small-scale datasets, especially compared to the state-of-the-art methods, which indicates that the generated texts greatly optimize the training of the classifier. We can see classification methods that perform well on large-scale datasets lose their advantages on small-scale datasets such as L-MIXED and DPCNN. The reason is that fewer samples lead to the overfitting of the deeper network. The unlabeled data provide useful information for semisupervised learning, so the semisupervised methods show good results on IMDB. As shown in Figure 7, the classification accuracy of various semisupervised models is similar, our method performs better than these semisupervised models on small-scale datasets. Note that, on the Part-1 dataset, the classification accuracy of Ad-training is slightly higher than that of the proposed method, but it uses a large amount of unlabeled real data in the training process. Besides, CCNN-GAN has the optimal or suboptimal performance on datasets of various sizes, which indicates that the method has a strong generalization ability.

To observe the experimental effect more conveniently, we bold the optimal experimental data. As shown in Tables 4 and 5, experimental results indicate that the accuracy of the classifier improves with the increase of training data. When the dataset is small, our method has better performance than other methods, but its accuracy improves more slowly as the dataset size increases. The reason is that the generation of textual features further enriches the dataset, which indirectly expands the dataset and reduces the importance of the original dataset. Besides, the classification accuracy is close to a saturation state when the dataset expands to a certain extent.

Overall, the experimental results show that the classifier based on deep neural networks achieves excellent performance in multiclass text classification. Although the proposed method is less competitive than the state-of-the-art methods on some large-scale datasets, CCNN-GAN is better than all the comparison methods in various small-scale datasets. Besides, our method inherits the advantages of the previous character-level convolutional network and makes it easier to adapt to multiple languages by updating the alphabet freely.

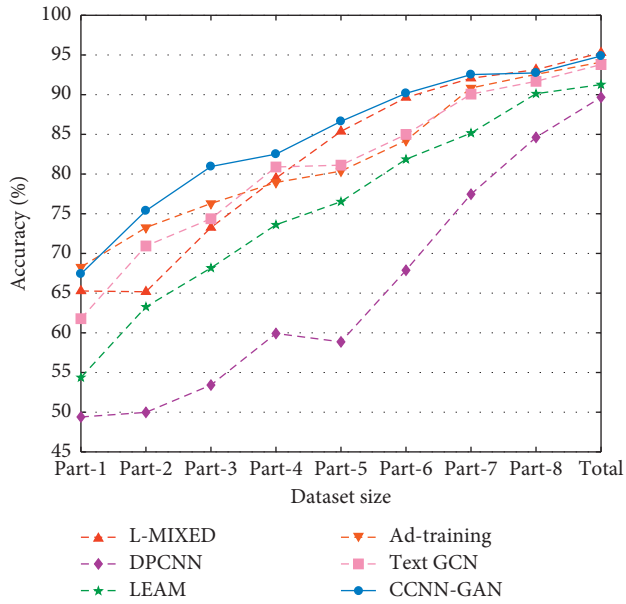

Figure 7: The performance on IMDB semisupervised classification task.

Table 4: Test performance on the AG-news classification task.

Method	Dataset size								
	Part-1	Part-2	Part-3	Part-4	Part-5	Part-6	Part-7	Part-8	Total
Tree-LSTM	37.96	58.13	68.33	76.20	87.12	90.75	91.56	91.80	91.83
Self-Attentive	30.81	54.89	72.26	85.47	86.37	90.83	91.28	92.03	91.17
Emb-CNN	48.83	68.38	79.74	84.11	83.85	85.94	87.93	89.36	90.08
Char-CNN	52.97	72.14	78.93	84.46	85.85	88.02	87.36	87.75	87.28
Char-CRNN	52.47	68.36	75.29	85.17	84.78	90.26	91.53	91.26	91.44
FastText	53.36	68.48	75.48	80.83	85.37	91.05	91.25	91.54	91.51
L-MIXED	24.68	25.45	66.21	79.25	84.20	90.27	**93.82**	**93.90**	**94.38**
DPCNN	25.94	25.62	69.85	77.29	82.44	86.71	90.89	91.12	93.13
LEAM	33.24	49.13	61.85	70.21	80.97	87.38	91.75	91.82	92.45
Ad-Training	47.29	57.63	69.15	76.21	82.37	87.83	90.89	90.22	91.37
Text GCN	57.32	68.96	77.83	84.54	87.62	89.35	90.74	91.32	92.56
CCNN-GAN	**66.47**	**77.53**	**85.37**	**86.93**	**88.34**	**91.48**	91.25	91.79	91.94

Table 5: Test performance on the DBPedia and 20NG classification task.

Method	DBPedia				20NG			
	Part-1	Part-4	Part-7	Total	Part-1	Part-4	Part-7	Total
Tree-LSTM	74.56	94.17	98.39	98.24	46.27	63.81	79.66	81.68
Self-Attentive	73.58	93.24	97.24	98.13	35.57	57.85	82.94	86.57
Emb-CNN	80.28	96.38	98.24	98.56	54.02	67.85	82.17	83.29
Char-CNN	76.86	**97.38**	98.24	98.37	52.53	73.19	81.28	82.61
Char-CRNN	80.12	96.27	**98.50**	98.67	53.95	67.38	81.72	83.78
FastText	82.68	95.98	98.45	98.63	44.32	67.29	**85.64**	87.26
L-MIXED	7.61	75.24	93.28	99.09	5.24	72.18	85.34	86.94
DPCNN	7.24	77.72	91.25	99.12	5.27	63.55	76.51	82.42
LEAM	79.65	95.26	97.38	99.02	49.16	64.81	83.67	87.13
Ad-Training	82.34	97.32	98.37	**99.24**	43.70	68.27	64.85	86.29
Text GCN	81.76	96.85	98.30	99.07	53.74	74.32	83.35	87.15
CCNN-GAN	**84.84**	97.35	98.47	98.64	**56.79**	**76.28**	84.32	**88.14**

5. Conclusion

In this paper, we propose a hybrid neural network framework for text classification. Our framework introduces generative networks to enrich corpus and utilizes a character-level convolutional network to extract latent semantic. Experimental results show that the performance of the framework on large-scale datasets outperforms other mainstream methods, and it performs significantly better than other methods on small-scale datasets. In the future, we intend to improve the output of the generative network and further enrich the generated text semantics.

Data Availability

The data supporting this paper are from the reported studies and datasets in the cited references.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work was partially supported by the National Natural Science Foundation of China (nos. 61702310 and 61772322) Major Fundamental Research Project of Shandong, China (no. ZR2019ZD03), and Taishan Scholar Project of Shandong, China (no. ts20190924).

References

[1] Z. Wu, H. Zhu, G. Li et al., "An efficient wikipedia semantic matching approach to text document classification," *Information Sciences*, vol. 393, pp. 15–28, 2017.

[2] T. Wang, L. Liu, N. Liu, H. Zhang, L. Zhang, and S. Feng, "A multi-label text classification method via dynamic semantic representation model and deep neural network," *Applied Intelligence*, pp. 1–13, 2020.

[3] F. Shang, H. Zhang, L. Zhu, and J. Sun, "Adversarial cross-modal retrieval based on dictionary learning," *Neurocomputing*, vol. 355, pp. 93–104, 2019.

[4] F. Shang, H. Zhang, J. Sun, and L. Liu, "Semantic consistency cross-modal dictionary learning with rank constraint," *Journal of Visual Communication and Image Representation*, vol. 62, pp. 259–266, 2019.

[5] M.-F. Ge, C.-D. Liang, X.-S. Zhan, C.-Y. Chen, G. Xu, and J. Chen, "Multiple time-varying formation of networked heterogeneous robotic systems via estimator-based hierarchical cooperative algorithms," *Complexity*, vol. 2020, Article ID 8357428, 18 pages, 2020.

[6] X. Chen, D. Li, P. Wang, X. Yang, and H. Li, "Model-free adaptive sliding mode robust control with neural network estimator for the multi-degree-of-freedom robotic exoskeleton," *Complexity*, vol. 2020, Article ID 8327456, 10 pages, 2020.

[7] X. Li, "Further analysis on uniform stability of impulsive infinite delay differential equations," *Applied Mathematics Letters*, vol. 25, no. 2, pp. 133–137, 2012.

[8] X. Li, T. Caraballo, R. Rakkiyappan, and X. Han, "On the stability of impulsive functional differential equations with infinite delays," *Mathematical Methods in the Applied Sciences*, vol. 38, no. 14, pp. 3130–3140, 2015.

[9] X. Li, J. Shen, H. Akca, and R. Rakkiyappan, "LMI-based stability for singularly perturbed nonlinear impulsive differential systems with delays of small parameter," *Applied Mathematics and Computation*, vol. 250, pp. 798–804, 2015.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *International Conference on Neural Information Processing Systems*, pp. 2672–2680, Montreal, Canada, December 2014.

[11] M. Zhang, J. Li, H. Zhang, and L. Liu, "Deep semantic cross modal hashing with correlation alignment," *Neurocomputing*, vol. 381, pp. 240–251, 2020.

[12] L. Liu, B. Zhang, H. Zhang, and N. Zhang, "Graph steered discriminative projections based on collaborative representation for Image recognition," *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24501–24518, 2019.

[13] L. Liu, S. Chen, X. Chen, T. Wang, and L. Zhang, "Fuzzy weighted sparse reconstruction error-steered semi-supervised learning for face recognition," *The Visual Computer*, pp. 1–14, 2019.

[14] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[15] T. Mikolov, *Statistical Language Models Based on Neural Networks*, vol. 80, Brno University of Technology, Brno, Czechia, 2012.

[16] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, https://arxiv.org/abs/1607.01759.

[17] Y. Kim, "Convolutional neural networks for sentence classification," 2014, https://arxiv.org/abs/1408.5882.

[18] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, https://arxiv.org/abs/1404.2188.

[19] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 649–657, Montreal, Canada, 2015.

[20] H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?" in *Proceedings of the Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, February 2018.

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modelling," 2014, https://arxiv.org/abs/1412.3555.

[22] X. Zhu, P. Sobihani, and H. Guo, "Long short-term memory over recursive structures," in *Proceedings of the International Conference on Machine Learning*, pp. 1604–1612, Lille, France, July 2015.

[23] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, February 2018.

[24] J. Wei and K. Zou, "Eda: easy data augmentation techniques for boosting performance on text classification tasks," 2019, https://arxiv.org/abs/1901.11196.

[25] Z. Wang and Q. Wu, "An integrated deep generative model for text classification and generation," *Mathematical Problems in Engineering*, vol. 2018, Article ID 7529286, 8 pages, 2018.

[26] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, https://arxiv.org/abs/1411.1784.

[27] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: sequence generative adversarial nets with policy gradient," in

Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, February 2017.

[28] J. Xu, X. Ren, J. Lin, and X. Sun, "DP-GAN: diversity-promoting generative adversarial network for generating informative and diversified text," 2018, https://arxiv.org/abs/1802.01345.

[29] Y. Li, Q. Pan, S. Wang, T. Yang, and E. Cambria, "A generative model for category text generation," *Information Sciences*, vol. 450, pp. 301–315, 2018.

[30] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2016, https://arxiv.org/abs/1605.07725.

[31] A. Ahamad, "Generating text through adversarial training using skip-thought vectors," 2018, https://arxiv.org/abs/1808.08703.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1097–1105, Lake Tahoe, CA, USA, 2012.

[33] J. Lehmann, R. Isele, M. Jakob et al., "DBpedia—a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[34] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015, https://arxiv.org/abs/1503.00075.

[35] Z. Lin, M. Feng, and C. N. d. Santos, "A structured self-attentive sentence embedding," 2017, https://arxiv.org/abs/1703.03130.

[36] Y. Xiao and K. Cho, "Efficient character-level document classification by combining convolution and recurrent layers," 2016, https://arxiv.org/abs/1602.00367.

[37] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Revisiting LSTM networks for semi-supervised text classification via mixed objective function," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6940–6948, 2019.

[38] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 562–570, Vancouver, Canada, July 2017.

[39] G. Wang, C. Li, W. Wang et al., "Joint embedding of words and labels for text classification," 2018, https://arxiv.org/abs/1805.04174.

[40] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7370–7377, 2019.