IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 16 August 2021, accepted 24 October 2021, date of publication 12 November 2021, date of current version 5 July 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3127913

D3-SACNN: DGA Domain Detection With Self-Attention Convolutional Network

KEJUN ZHAO[1,3], WEI GUO[2], FENGLIN QIN[3], AND XINJUN WANG[2]
[1]School of Computer Science and Technology, Shandong University, Qingdao 266237, China
[2]School of Software, Shandong University, Jinan 250101, China
[3]Informatization Office, Shandong University, Jinan 250100, China

Corresponding author: Wei Guo (guowei@sdu.edu.cn)

This work was supported in part by the National Key Research and Development Project of China under Grant 2019YFB1705904; in part by the National Nature Science Foundation of China under Grant 91846205, Grant 61772316, and Grant 61907026; in part by the Innovation Methods Work Special Project under Grant 2020IM020100; in part by the Science and Technology Development Plan Project of Shandong Province under Grant 2019JZZY020505; in part by the Key Research and Development Project of Shandong Province under Grant 2019GGX101009; in part by the Shandong-Chongqing Technological Collaboration Plan under Grant cstc2020jscx-lyjsAX0010; and in part by the Project of Shandong Province Higher Educational Science and Technology Program under Grant J18KA392.

ABSTRACT Botnets are currently one of the main cyber security threats. In order to enhance the concealment, botnets usually use Domain Generation Algorithm (DGA) to establish communication between bots and command and control servers. Character-based deep learning methods are widely researched in the classification of DGA domains to detect botnets and have achieved good results. But the pronounceable DGA domain detection is still a challenge, since the linguistic statistical characteristics of the pronounceable DGA domains and benign domains are very similar. We propose a multi-head self-attention convolutional network method for DGA domain classification task. We use a shallow convolutional neural network to extract hidden features of domain characters. The multi-head self-attention mechanism with different input values is used to effectively obtain the relationship between the characters and the extracted implicit features, which will help us more effectively distinguish between pronounceable DGA domains and benign domains. Experiments on public data show that our model can effectively detect various types of DGA domains. Especially for the pronounceable DGA domains, our method is significantly better than other detection methods.

INDEX TERMS Convolutional network, domain classification, domain generation algorithm, self-attention.

I. INTRODUCTION

Botnet is one of the most serious network security threats today. It can launch a variety of network attacks, such as DDoS attacks, spam distribution, data theft encryption, and unauthorized digital currency mining [1]. A typical botnet usually consists of a command and control (C2) server and controlled bots. The C2 server sends malicious attack instructions and software updates through the network. In the early botnets, to facilitate the communication between the bots and the C2 server, the IP address of the C2 server was generally hard coded into the malicious program. This method has obvious shortcomings. Once the malicious program is decompiled or C2's IP address is discovered through traffic analysis, the IP will be added to the blacklist and the controller will face the risk of losing control of the entire botnet.

To reduce the risk of direct exposure of the C2 host IP, botnets usually use a Domain Fluxing [2] technology to dynamically switch the domain of the C2 server. The botnet controller will pre-configure multiple domains in the malware of the botnet, and then randomly select a domain to register and point to the C2 server. The bot initiates DNS queries on all domains until the registered domains are resolved. By obtaining the registered domain, the bot can obtain the IP address of the C2 server, and then establish a communication channel with the C2 server. With this technology, the botnet tries to hide the C2 server by changing the IP address. A simple and effective defense method is to preemptively register all known malicious domains and direct them to a host or black hole address [3]. The premise of this method is to obtain a list of domains pre-configured by the botnet.

Botnets currently use domain generation algorithms (DGA) to avoid using a static list of domains. A typical DGA dynamically generates a large list of pseudo-random domains

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

periodically based on agreed seeds (such as time and date, popular search strings on social networks and headlines, etc.), and the size of each list generated is usually very large. For example, *Conficker* C can generate 50,000 domains every day, and only a very small number of domains are registered. In a DGA botnet, bots only need to perform domain queries to obtain the address of the C2 server. The C2 server can be well hidden in these pseudo-random domains, which greatly increases the difficulty of detection.

Security researchers early used reverse engineering [4] to obtain the DGA domains from the botnet code. These kinds of methods are difficult to implement, and the attacker can update the DGA algorithm remotely. Since malware using DGA domains needs to initiate a large number of DNS queries, DNS traffic analysis is adopted for detection [5]–[8]. However, this type of technology generally needs data from a large time window, so it is difficult to use for real-time detection and prevention.

DGA domains have added randomness to prevent conflicts with legal domains, which results in statistical differences between general DGA domains and benign domains. Therefore, classification detections from the perspective of domain characters have become popular methods [9]–[11]. They usually use manual methods to collect multiple characteristics of domain characters, including length, entropy information, phonetic-consonant ratio, n-gram information, etc., and build a machine learning classification model to distinguish DGA domains from benign domains. Feature extraction is a difficult task requiring a lot of statistical analysis. Furthermore, it is often necessary to extract new features for newly emerging DGA families, which makes it difficult for this type of method to find new DGA domains.

Deep learning [12] has the characteristics of large model capacity and no need for manual feature extraction and has achieved very significant results in various application scenarios such as natural language processing and image recognition. DGA domain detection based on deep learning has become the latest hot spot method. The advantage is that there is no need to manually extract features and high accuracy. Researchers classified domains based on recurrent neural networks (for example, long and short-term memory network LSTM [13]) and achieved good results [14]–[16]. Since recurrent neural networks need to rely on the results of the previous step, it is difficult to use GPU hardware for parallel acceleration, so the training and detection speed is slow. Studies have shown that convolutional neural networks(CNN) perform well in serialized character data [17], [18]. CNN-based networks are also used for DGA domain detection [19], [20].

In recent years, more and more DGA algorithms start generating pronounceable domains, and these malicious domains are more similar to benign domains. For example, the "middleapple.net" domain generated by *suppobox* DGA is difficult to identify as a DGA domain without any other information. The above conventional domain detection methods rely on the significant language characteristics differences between the DGA randomly generated domains and the benign domains. So the detection accuracy of those pronounceable DGA domains is low. Researchers have proposed some solutions, such as the pre-trained context-sensitive word embedding technology [21] and semantic representation technology [22].

In this paper we propose D3-SACNN, a method based on multi-head self-attention convolutional networks, for DGA domain classification. This method aims to enhance the ability to detect pronounceable DGA domains while still maintaining high accuracy against random domain detection. It needs to process only the character data of the domain itself. D3-SACNN avoids manual feature extraction and can adapt to various types of DGA domains. In the D3-SACNN model, a shallow convolutional neural network is used to extract the hidden features of the domain characters. We use a multi-head self-attention mechanism with different input values in order to obtain better interdependence between features. The attention mechanism assigns different weights to feature information to obtain more important and critical information for the task. Experiments show that the self-attention mechanism can effectively obtain the correlation between the characters and the extracted implicit features. For example, benign domains usually contain multiple words related in context, while DGA domains lack this type of association due to their randomness. In the DGA domain classification task, this will help us more effectively distinguish between pronounceable DGA domains and benign domains.

The main contributions of this paper are as follows:

- We propose a lightweight method for DGA domain detection and prediction, D3-SACNN. We introduce the self-attention mechanism into the model to obtain the interrelationships between features more effectively.
- Experiments prove that D3-SACNN can effectively distinguish DGA domains from benign domains. Especially for the pronounceable DGA domain, it has very good results.
- In the evaluation tasks of two-classification, multi-classification, and leave-class-out (for the model's ability to detect new DGA domain families), the D3-SACNN method is overall better than the other three deep learning methods compared.

In the next section, we will present the related work. The detail of the proposed method is discussed in section III. We present the data sets, method evaluations, full results and analysis in section IV. The last section summarizes our research and introduces future work.

II. RELATED WORK

Botnets send DNS queries at the same time as group activities. Anomaly-based detection methods can distinguish botnet group query activities from legitimate DNS traffic, thereby discovering botnets and their malicious domains. Lee *et al.* [23] checked the DNS query information of the clients accessing the known malicious domain to find the unknown malicious domains via looking through the order relationship and statistical commonality with the known malicious domain. Bilge *et al.* [24] extracted 15 characteristics in four categories of time attribute, DNS response information, TTL information and domain information from DNS query traffic, and then identified the query operations of botnets based on the decision tree algorithm. Grill *et al.* [25] proposed a statistical method that uses collected NetFlow/IPFIX statistical data to detect DGAs. Kwon *et al.* [26] utilized PSD (Fourier Transform) signal processing technology and spectral density measurement technology to analyze user domain query behavior and detect illegal domains based on the time sequence relationship. At the same time it improved the processing speed.

When trying to contact the C2 server, the DGA botnet usually generates a large number of domains to avoid the blacklist detection mechanism, but only one or a small minority of domains will be registered. Therefore, the infected host will initiate many failed domain (NXDOMAIN) queries. Many studies have used this feature and achieved good results. Reference [5] found abnormal domains based on the frequency and time concentration of the domain queries. The method of detecting abnormal recurrences NXDOMAIN is very effective. Yadav *et al.* [6] used the failed domain information to detect the registered DGA domains, by studying the time correlation and information entropy characteristics of the successful domains versus the failed domains. DBod [7] found that hosts infected with the same DGA malware have the same set of query domains, and most of the queries failed. The author uses a clustering method to distinguish the clusters attacked by botnets from ordinary clusters. Antonakakis *et al.* [8] used a clustering algorithm to perform clustering based on the similarity of the domain composition and the machine groups that query these domains, and then assigned the generated clusters to known DGA botnet models. If a certain cluster obtained by clustering cannot be assigned to a known model, then a new type of botnet can be discovered. Lei *et al.* [27] tried using bipartite graphs to analyze DNS traffic and time series patterns of DNS queries, then adopted graph embedding technology to characterize domains, and finally identified malicious domains with the SVM classification algorithm. NXDOMAIN traffic is much smaller than the entire DNS query traffic. Therefore, the detection method based on NXDOMAIN features is faster and the model is lighter. However, these kinds of methods require statistical information in a larger time window and cannot achieve real-time detection.

Malicious domains programmatically generated by DGA usually contain random characters to avoid conflicts with benign domains. Classification based on the semantic features of domains has become an important research direction. Yadav *et al.* [2], [28] analyzed the information entropy of the distribution of alphanumeric unigrams and bigrams in domains and divided the training set into two subsets: the subset generated by DGA and the benign domain subset. The classification is carried out in batches. Each batch of unknown domains is clustered by shared secondary domains and domains with the same IP address, and then the unigram and bigram distributions of each cluster are calculated. Different distances including Kullback-Leibler (KL) distance are used to compare with the two known subsets. The Phoenix system [9] uses two basic language features: meaningful character ratio and n-gram normal score. Meaningful character ratio calculates the ratio of characters containing meaningful words in domains. Unknown domains are classified based on the characters of the domain and the Mahalanobis distance of the query. Once the domain is classified as DGA, they are sent to the DBSCAN clustering method for further classification. Tong *et al.* [10] improved the Phoenix algorithm and used more domain character features such as information entropy, frequency of n-grams and Mahalanobis distance for domain classification to detect DGA domains. Truong *et al.* [11] analyzed many benign domains and DGA domains and found that the characters of the domains have obvious statistical deviations. Based on the length of domains and the characteristics of character information entropy, they constructed a J48 decision tree classification algorithm to distinguish between benign domains and DGA domains. The above research needs to manually extract features, and these features have insignificant statistical information for different types of domains. Therefore, new features usually need to be extracted for different types of DGA domains. The feature extraction is a difficult task, and it is difficult to ensure that satisfactory features can be obtained. In addition, based on the limited extracted features, the accuracy of the above DGA domain detection method is relatively low.

Deep learning [12] has achieved significant results in computational vision and natural language processing. The classification of DGA domains based on deep learning algorithms has become the current mainstream research method. This type of method mainly uses recurrent neural networks and convolutional neural networks based techniques to extract features on domain datasets without manually selecting feature information. LSTM can learn the context between characters well, so this method can classify DGA domains without manually extracting any features. Woodbridge *et al.* [14] first used deep learning technology to classify domains, using word embedding technology to vectorize domains, and then classifying domains based on LSTM. Curtin *et al.* [15] proposed the smashing score as a measure of the similarity between domains and English words. The smashing score is the average n-gram overlap rate (n is in the range of 3-5) between the domains and the words in the English dictionary. The author also used WHOIS query information to assist LSTM classification. This method achieves better results in the dictionary DGA classification. Due to the category imbalance problem in DGA training samples, Tran *et al.* [16] proposed a cost-sensitive LSTM.MI algorithm. It introduced the cost term into the back propagation mechanism of LSTM to improve LSTM's robustness to imbalance problems. The method using a cyclic neural network LSTM can well find the long-distance dependence between the characters of the domain, and the accuracy rate also well exceeds the traditional method based on character statistics. However, because the recurrent neural network relies on the calculation results of the previous step and cannot fully utilize GPUs to achieve large-scale parallel operations, the training speed and detection speed are slow. CNNs are increasingly used in natural language processing [17], [29]. Recently, Huang *et al.* [19] used n-gram to vectorize domains, and classified DGA domains with multi-layer res-CNN.

To avoid detection more and more DGA algorithms have started generating pronounceable domains which are similar to benign domains. Most of the conventional domain detections mentioned above rely on the significant differences in language characteristics between the DGA randomly generated domains and the benign domains. So it is difficult to effectively detect the pronounceable domains. Researchers tried to use some natural language processing techniques to enhance the pronounceable DGA domains detection ability. Koh and Rhodes [21] used pre-trained context-sensitive word embedding technology. This method can use only a small training set to get results. Xu *et al.* [22] used the semantic method (n-gram) to extract the interrelationships between the characters of the domain and completed the domain classification through a convolutional network.

Bahdanau *et al.* [30] introduced an attention mechanism in machine translation tasks to achieve excellent results. This mechanism has quickly become a research hotspot in various fields of NLP, including machine translation [31], question and answer systems [17], and text summaries [32]. The attention mechanism is inspired by human perception, giving more weight to important information, and improving the model's ability to acquire key information. Vaswani *et al.* [33] proposed the concept of self-attention, which became the main direction of attention research. In the process of feature extraction, self-attention can obtain the relationship between features at the same level without being limited by distance.

III. MODEL

Our proposed DGA domain classification method D3-SACNN is based on the multi-head self-attention convolutional network. Figure 1 shows the model architecture. First, the domain characters are converted into embedding vectors, and the features are extracted through the CNN module. Then the self-attention part is responsible for obtaining long-distance dependencies, and finally it uses the classification function for classification. For domain classification tasks, it is very important to obtain the contextual relationship between the characters of a domain. The receptive field of a convolutional network is a fixed grid. So the output information is only related to the input information in the grid. These make it difficult for the convolutional network to obtain long-distance context relationships. To obtain a wider range of dependencies, the convolutional network must increase the size of the grid or the depth of the model, which leads to a greatly increased amount of model parameters. To overcome the disadvantage of CNN, we use a multi-head self-attention mechanism [33]. Different from the traditional attention mechanism, self-attention directly obtains the relationship between features and other features, and the process of feature extraction and context acquisition are processed in a unified manner. This mechanism has been proved to be able to effectively calculate the long-distance dependence of features. This is mainly because the output of self-attention is the result of the weighted summation of all inputs, that is to say, the receptive field of self-attention is all the input information.

A. EMBEDDING LAYER

As we know, a domain is a string used to identify Internet computers. The domain is separated into multiple parts by the character '.': from right to left are top-level domain OR first-level domain, second-level domain, third-level domain, and so on. For example, in the domain 'www.google.com,' 'com' is the top level domain, 'google' is the second level domain, and 'www' is the third level domain. The domains generated by DGA are mainly second-level domains, so we use the second-level domain as the input of the model. Unless otherwise specified, the following domains will specifically refer to second-level domains. We first need to vectorize domain characters as input to the model. In natural language processing, character vectorization methods mainly include N-gram coding, one-hot coding, and distributed representation, namely embedding. N-gram encoding uses a sliding window of N characters to generate character sequences. This method retains the context of characters and is often used in shallow natural language models. A large N value causes the vector space dimension to be too large, which could lead to the problem of data sparseness, and thus normally we avoid using a large N value. One-hot encoding generates a character dictionary. The encoding of each character is represented by a vector, and the length of the vector is the size of the dictionary. The sequence position of the character in the vector is 1 in the dictionary, and the other values are 0. The sparse One-hot encoding vector cannot effectively express the relationship between characters. Our method uses distributed representation to embed characters into a real vector space.

Specifically, according to RFC-1035 [37], the domain consists of ASCII letters, numbers, and the symbol '-.' We use '?' to represent any other characters. And thus we have the domain set of characters C with a total number of 38 characters.

For any given domain D $(c_1, c_2, \ldots, c_i, \ldots, c_n)$ it contains a sequence of characters from set C (i.e. $c_i \in C$). We first convert the domain characters into their corresponding index values (starting from 1) in set C. To keep the input length consistent, we set the length $l = 64$ for all domains. When the actual length of the domain is less than 64, we pad it with 0. If the actual length of the domain is greater than 64, it is truncated to the first 64 characters, which is a rare case. The indexed domain is transformed into a character vector sequence through the word embedding layer $\boldsymbol{w}_{1:l} = \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_l$, where $\boldsymbol{w}_i \in ?^d$ and d is the embedding layer

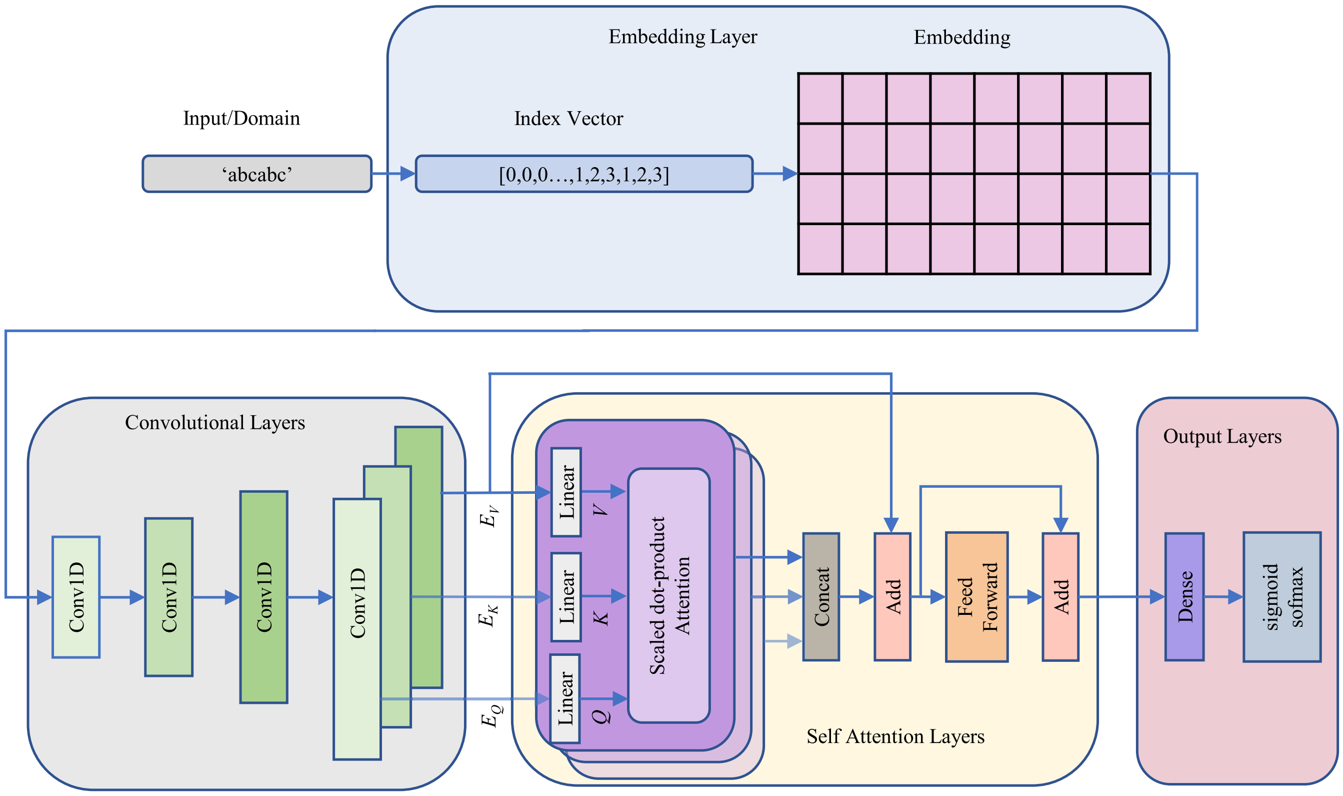

FIGURE 1. **Architecture of the proposed D3-SACNN model.**

vector dimensions. Parameter d is set to 128 in this paper, which retains sufficient feature context information for us. The embedding vectors of the characters are obtained through the model training process, so that we can obtain more hidden relationships between characters.

B. CNN LAYERS

This paper uses one-dimensional convolution that slides a convolution kernel with width of k on the domain sequence embedding vectors. Each time a CNN layer performs convolution calculations over the k input sequences:

$$\boldsymbol{x}_i = \oplus\left(\boldsymbol{w}_{i:i+k-1}\right), \tag{1}$$

$$c_i = g\left(\boldsymbol{x}_i \cdot \boldsymbol{u} + b\right), \quad \boldsymbol{x}_i, \boldsymbol{u} \in \mathbb{R}^{k \cdot d}, \ c_i, b \in \mathbb{R} \tag{2}$$

Operator $\oplus\left(\boldsymbol{w}_{i:i+k-1}\right)$ represents the concatenating vector $\boldsymbol{w}_i, \ldots, \boldsymbol{w}_{i+k-1}$, $\boldsymbol{u}$ is the weight vector in the convolution kernel, and b is the bias variable. The convolution calculate the inner production of the weight vector $\boldsymbol{u}$ in the filter and vector $\boldsymbol{x}_i$, plus the offset result b, and then the result is passed through a nonlinear function g and output. The nonlinear function g adopts the ReLU function in our model. Each layer uses n filters $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ to form a filter matrix $\boldsymbol{U}$ to obtain the corresponding output $\boldsymbol{c}_i$:

$$\boldsymbol{c}_i = g\left(\boldsymbol{x}_i \cdot \boldsymbol{U} + \boldsymbol{b}\right), \quad \boldsymbol{x}_i \in \mathbb{R}^{k \cdot d}, \ \boldsymbol{c}_i, \boldsymbol{b} \in \mathbb{R}^n, \ \boldsymbol{U} \in \mathbb{R}^{k \cdot d \times n} \tag{3}$$

This paper uses a four-layer convolutional network, containing 128, 256, 256 and 512 filters successively, and the filter width k is 3, 5, 7 and 9 respectively. Experimental results show that gradually increasing the width of the filter can expand the field of view of the convolution operation and obtain better results. The last layer uses 3 convolutional blocks of the same size to generate input for the self-attention layer.

C. SELF-ATTENTION LAYERS

As shown in figure 1, an important unit in the self-attention mechanism is scaled dot-product attention. The input of the scaled dot-product attention is composed of a series of (Query, Key, Value) vectors. The dot product is used to calculate the correlation between a certain Query and all Keys, and the attention weight of the Value corresponding to the Key is obtained. To improve the calculation efficiency, all (Query, Key, Value) vectors are converted into matrices Q, K and V during the calculation process for matrix calculation. Different from attention cells [33] using the same input to generate Q, K and V, we use different input E_q, E_k and E_v for generating Q, K and V. The purpose is to obtain more feature information for attention calculation. For example, the role of Q and K is to obtain the weight relationship between input, and their characteristics may be different from the characteristics of V [30], [34]. E_q, E_k and E_v are linearly converted to Q, K and V. Calculation formula is as follows:

$$Q = E_q W_q \tag{4}$$

$$K = E_k W_k \tag{5}$$

$$V = E_v W_v$$

$$\{E_q, E_k, E_v\} \in \mathbb{R}^{l \times d_{model}}$$

$$\{W_q, W_k, W_v\} \in \mathbb{R}^{d_{qkv} \times d_{model}} \tag{6}$$

The attention matrix is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{7}$$

Firstly Q and the transposition of K perform dot-product calculation, and then normalize with the softmax function to get the attention weights. In order to prevent large value of the dot products from causing extremely small gradients of the softmax function, the dot products are scaled by $1\big/\sqrt{d_{qkv}}$ times. Finally, for each output, the attention value is the weighted sum of all Keys.

In order to obtain more subspace information, we use multi-head attention, that is, multiple self-attention calculations are performed in parallel, and the final results are concatenated:

$$\begin{aligned} &\mathrm{MultiHead}(Q, K, V) \\ &\quad = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_i}, \ldots, \mathrm{head_h}) \\ \text{where } &\mathrm{head_i} = \mathrm{Attention}(Q_i, K_i, V_i) \end{aligned} \tag{8}$$

In this study, the experimental results show that better results can be obtained when the number of heads is 4, we choose the number of heads $h = 4$. In order to perform residual calculation with the input, the concatenated dimension must be consistent with the input dimension, with $d_{dkv} = d_{model}/h = 128$. Finally similar to the encoder structure in transformer, the output of the multi-head attention and the input E_v perform the residual operation [35], and then output through two linear transformations:

$$FFN(x) = max\,(0, xW_1 + b_1)\, W_2 + b_2 \tag{9}$$

D. OUTPUT LAYERS

In the output layers we use a 128-node fully connected layer to process the implicitly extracted features, and finally use the classification function to obtain the normalized probability estimation. For the two-class classification case, we use the sigmoid function as the classification function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{10}$$

The output of the sigmoid function is a real number, which represents the probability estimate that a sample belongs to DGA domains. We choose the cross-entropy function as the loss function:

$$L(\hat{y}, y) = -ylog(\hat{y}) - (1 - y)log(1 - \hat{y}) \tag{11}$$

where $\hat{y}$ is the predicted probability, y is the true category label of the sample. 1 represents a DGA domain, and 0 represents a benign domain.

For multi-class classification tasks, we use the softmax function as classification function:

$$p_i = \frac{\exp\left(z_{\mathrm{out}_i}\right)}{\sum_j \exp\left(z_{\mathrm{out}_j}\right)} \tag{12}$$

The function output p_i is the normalized probability of different classes. The predicted category k is the class with the highest probability:

$$k = \underset{i}{\mathrm{argmax}}\, p_i \tag{13}$$

The corresponding loss function is a cross-entropy function as well:

$$L = -\sum_{c=1}^{C} y_c \log\left(p_c\right) \tag{14}$$

where C is the number of classes, y_c is the value of the sample corresponding to different class c. When a sample belongs to a specific class, $y_c = 1$, for other classes $y_c = 0$. p_c is the predicted probability that the sample belongs to class c.

IV. EXPERIMENT AND ANALYSIS

We adopt supervised learning to evaluate the proposed algorithm based on public data in the experiments. There are mainly three experiments: binary classification, multi-class classification and leave-class-out classification. Binary classification is used to test the algorithm's ability to distinguish between benign domains and DGA domains. Multi-class classification is used to evaluate the distinguishing ability of different types of DGA domains, which can obtain specific DGA classifications in practical applications. The leave-class-out experiment is used to simulate the ability to recognize new DGA family domains. We first introduce the details of the data set, and then define the evaluation indicators of the experiment. After that, we provide the optimization results of the model structure and parameters. Finally, we conduct a comparative analysis with other works.

A. DATA SET

Experimental data includes a benign domain set and a DGA domain set. To facilitate experimental reconstruction and verification, all data are obtained from the public data set on the Internet. For the DGA domains, we used 360 DGA feeds [36], which can be accessed openly. The samples' numbers of DGA families are different. Some families have more than 80,000 samples, while some others have only dozens of samples. To reduce the impact of unbalanced data on the model, we used the following strategies to sort out DGA domains: for DGA families with more than 40,000 samples, 40,000 samples are randomly selected. DGA families with less than 1,000 samples are discarded. The chosen DGA families are all put into the data set. After applying the above preparation strategy we have 21 DGA families with each family's sample number in interval of [1000, 40000] and a total of 305,960

TABLE 1. Experimental data details.

DGA family	pronounceable	Min-length	Max-length	Supported-num	Sample
pykspa_v1	N	6	15	40000	aaamiaiq
murofet	N	11	44	4781	yhsqkploqshhjwbl
ramnit	N	8	19	18814	imboajaksftmyk
gameover	N	16	31	11998	1ld7ywilejzvmktu6ve14rs962
ranbyus	N	14	17	10593	tfholjtspefhoxope
locky	N	5	17	1158	urmrhedeslqtg
simda	**Y**	5	11	21645	dikoniwudim
qadars	N	12	12	1800	dez0127whmbc
symmi	**Y**	8	15	4256	anboinkut
banjori	**Y**	7	26	40000	earnestnessbiophysicalohax
tinba	N	9	12	40000	nvfowikhevmy
rovnix	N	18	18	40000	h53bxo2qz45n6io7um
dyre	N	34	34	1000	s736c6cf736c08d9f8e9ea2fbc801d1052
suppobox	**Y**	8	19	1746	subjecttravel
chinad	N	16	16	1000	hi7qlt2mm75pnuuy
necurs	N	3	21	7918	kqeykmhngmfsix
cryptolocker	N	12	15	1000	eliiydcwbevnnj
shiotob	N	8	15	5989	9g2rdi9uga
shifu	N	7	7	2541	nqqxqdg
emotet	N	16	16	40000	affvqugewqpbcbic
virut	N	6	6	9721	yeftgm

samples. We selected the first 305,960 domains from the Alexa top 1 million domains as the benign domain set. Alexa provides a ranking of global website traffic. The domains with the top rankings are the more widely used domains.

Table 1 shows some statistical information of DGA families including the minimum length, maximum length of domains and the number of samples in each DGA family, as well as a sample example. In 7 DGA families (*qadars, rovnix, dyre, chinad, shifu, emotet* and *virut*) the length of the generated domains is fixed, while in the other remaining families the length is variable. There are 4 DGA families (*side, symmi, banjori* and *suppobox*) whose domains are pronounceable. To avoid domain detection, pronounceable DGA families construct DGA domains through word combinations or using natural language pronunciation features. This type of domain has similar consonants, vowels, and character distribution compared to benign domains. It can be imagined that the pronounceable domains are very difficult to detect, as shown from the experiment's results.

B. EVALUATION METRICS

In this article, the DGA samples in the binary classification experiment are positive samples, and the benign domains are negative samples. In a multi-class classification experiment, for each DGA family, the samples of this family are positive samples, and the others are negative samples. According to the relationship between the predicted values and the actual values, the following 4 possible results are included:

1. True Positive (TP) sample is positive, and prediction is positive.
2. True Negative (TN) sample is negative, and prediction is negative.
3. False Positive (FP) sample is negative, and prediction is positive.
4. False Negative (FN) sample is positive, and prediction is negative.

Precision, recall, and F1 are used to evaluate the classification performance, which are defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{15}$$

$$recall = \frac{TP}{TP + FN} \tag{16}$$

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{17}$$

where TP, TN, FP, FN represent the number of different prediction results. For positive samples, the precision rate is the proportion of true positive samples predicted by the

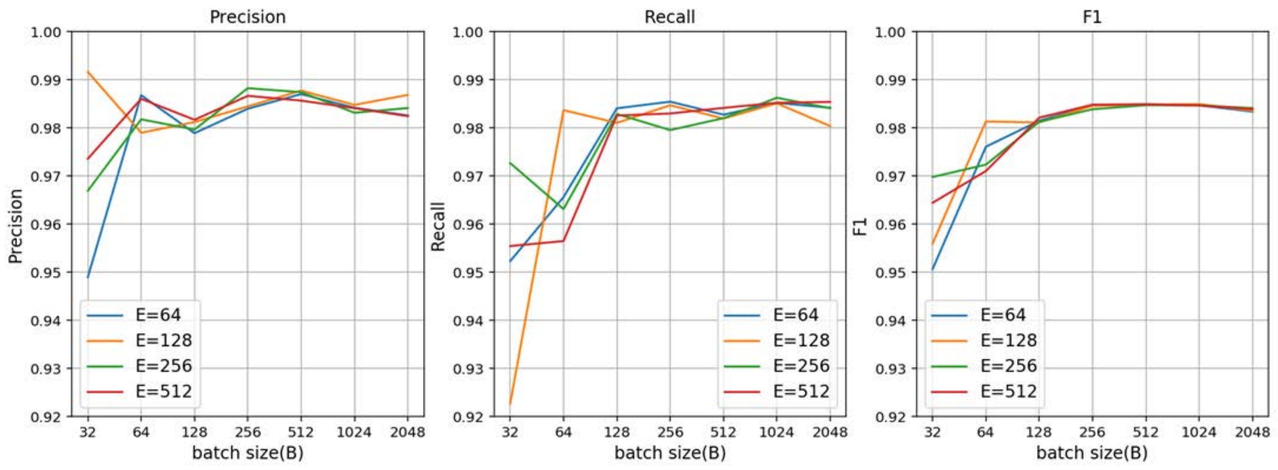

FIGURE 2. **Performance comparison with different batch sizes and embedding sizes.**

classifier as positive samples, and the recall rate is the proportion of positive samples predicted as positive samples. We hope that the accuracy and recall values of the model are as high as possible. However, there are mutual constraints between the two metrics. According to actual application scenarios, we may pay attention to different metrics. When we want to detect as many DGA domains as possible, the recall metric is more important. When fewer false alarms are needed, accuracy is a better indicator. To balance the impact of accuracy and recall, a more comprehensive F1-Score is introduced. High F1-Score normally indicates fewer extreme cases of accuracy and recall.

C. HYPER-PARAMETERS

All experiments are conducted on a server that has 2 Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz CPUs, 128GB of main memory, and a GeForce GTX 1080Ti GPU. The operating system is Ubuntu 16.04 x64, and the experiments are developed using Python language. It uses the scikit-learn machine learning library (v0.20.4) and TensorFlow platform (v1.4.1).

The method in this article involves the following important parameters:

1) PARAMETER B

Batch size. In the training process, the common practice is to divide the training set into multiple mini-batches in each epoch, and the number of samples in the mini-batch is batch size. The larger the batch size, the smaller the training time of a single epoch, and the smaller the gradient difference between batches, which is conducive to model convergence. But too large batch size also causes the gradient difference to be too small, and the loss value is easy to fall into a local minimum. If the batch size is too small, the more random the samples in the mini-batch, the more noise, and it is not conducive to convergence. We tried different batch size values, namely $B = 32, 64, 128, 256, 512, 1024, 2054$.

2) PARAMETER E

Embedding size. Embedding size is the vector dimension after character embedding and vectorization. If the Embedding Size is too small, it is difficult to obtain enough information. If it is too large, it will increase the calculation cost and may increase the risk of overfitting. We initialize the embedding size to 128, and then evaluate the effect of different values on the result, these values are 64, 128, 256, 512.

We use the result of the binary classification experiments to evaluate the parameters. The evaluation indicators are all the best results we got. Figure 2 shows the accuracy, recall and F1-score of different combinations of parameters B and E.

As shown in the figure, when $B < 256$, the values of precision are unstable, and the values of recall gradually increase. For example, when $B = 32$ and $E = 128$, Precision reaches the highest value of 0.9916, but at the same time the value of recall is only 0.9226. These mean that the accuracy rate of the DGA domain predicted by the model is high, but a large proportion of the DGA domains are not correctly predicted, resulting in a low recall value. When $B >= 256$, Recall and Precision tend to stabilize. In the F1 result graph, it shows a similar pattern. When the batch size is small, the samples show more randomness, and the trained prediction model is not stable enough. And thus, we need to choose a reasonably big batch size to ensure more stable results. In addition, when the batch size is greater than 512, the precision value has a downward trend, and when $B = 2054$ the F1 result also gets lower compared to smaller B value. It indicates that the larger B is not always better. When the batch size is 256, the precision and recall values have reached a relatively balanced state, and the F1 value has also reached the peak stage. As for Embedding size E, all the 3 graphs show that when the batch size is greater than 128, the difference between the results produced by different E is very small, indicating that the size of several E we tested has little effect on the results. Taken

TABLE 2. Performance results with different filter sizes.

Filter size	Precision	Recall	F1
3	0.9831	0.9860	0.9845
5	0.9836	0.9844	0.9840
7	0.9863	0.9805	0.9834
9	0.9851	0.9805	0.9828
[3,5,7,9]	0.9872	0.9834	0.9853

together we selected the following parameter combinations: B = 256, E = 128.

3) PARAMETER F

Filter size, the size of the convolution kernels. We tested two strategies: fixed filter size across all 4 convolutional layers, and variable filter size. For fixed filter size strategy, we tested 3, 5, 7, and 9. For variable filter size, we tried gradually increasing the filter size by setting [3, 5, 7, 9] as the filter size for each of the four layers respectively. Table 2 shows the binary classification results of different filter sizes. When the convolutional layers have the same filter size, the performance will decrease as the filter size increases. If the filter size is too big in the first few layers of the convolution, the calculated field of view is also large, and it can not extract subtle feature information enough. The F1 score of the increasing filter size layer by layer setting is 0.9853, which is better than the fixed filter size method. In the case of the same depth of the convolutional layers, the increased filter size can enlarge the acceptance range of the subsequent layer convolution. In this way, the context dependence of a larger range of features can be obtained, thereby improving the detection ability of the model. Therefore, we choose the strategy of gradually increasing the filter size of the convolutional layers, and the filter sizes are 3, 5, 7, and 9 respectively.

4) PARAMETER H

Multi-head attention head size. In the previous experiments, we set the head size H = 8 initially as in the Transformer model. We evaluated other values as well including 1, 2, 4, and 16. It should be noted here that since the output results need to perform a residual operation with the input values, the input and output dimensions of the attention are required to be consistent. Therefore, when adjusting the head size, the output vector dimension of each head in the attention block should also be adjusted accordingly to ensure that the input and output dimensions match. In multi-head attention, the more the number of heads, the more feature information can be obtained theoretically. At the same time, more heads increase the complexity of the model and may also increase the risk of model overfitting. It can be seen from Table 3 that the initialized 8 heads did not get the best results. When the head size = 4, better results can be obtained in this scenario. The head size is selected as 4.

5) PARAMETER D

Dropout rate. In order to reduce the over-fitting of the model we adopt Dropout strategy [37]. We evaluated the effects on

TABLE 3. Performance results with different filter multi-head.

Number of head	Precision	Recall	F1
1	0.9856	0.9849	0.9852
2	0.9868	0.9844	0.9856
4	0.9852	0.9866	0.9859
8	0.9872	0.9834	0.9853
16	0.9881	0.9825	0.9853

TABLE 4. Performance results with different dropout rates.

Dropout rate	Precision	Recall	F1
0.0	0.9852	0.9866	0.9859
0.1	0.9868	0.9859	0.9863
0.2	0.9863	0.9851	0.9857
0.3	0.9883	0.9829	0.9856
0.4	0.9865	0.9843	0.9854
0.5	0.9862	0.9845	0.9853

TABLE 5. Hyper-parameter configuration.

Hyper-parameter	value
Batch size	256
Embedding size	128
filter size	[3,5,7,9]
head size	4
Dropout rate	0.1

over-fitting of different dropout rates including 0 (i.e. dropout is disabled), 0.1, 0.2, 0.3, 0.4 and 0.5. As shown in Table 4 when the dropout is 0.1, the detection result of the model is improved, and the F1 value reaches 0.9863. When the dropout gradually increases after 0.1, the detection results of the model decrease instead. It shows that dropout discards too many neurons and weakens the model's fitting ability. According to the experimental results, we choose the dropout rate to be 0.1.

In summary, the main parameters we selected are:

D. RESULTS & ANALYSIS

We will compare with several existing methods based on deep learning:

LSTM [14]: The research was the first one applying deep learning to DGA domain detection. The author used Long Short Memory Network (LSTM) to extract features of domains. First, the domain characters are vectorized, and then the hidden features are extracted from the character vectors through LSTM serialization, and finally the sigmoid and softmax function are used for classification. The result of this method is stronger than the traditional methods based on manual feature extraction.

Dilated-CNN [20]: This study used a convolutional network for DGA domain detection. To increase the receptive field of the convolutional network, the study introduced a dilated causal convolutional layer, and used residual

TABLE 6. Binary classification performance comparison of different algorithms.

	Precision	Recall	F1-Score
LSTM	0.9852	0.9784	0.9818
Dilated-CNN	0.9857	0.9801	0.9829
n-CBDC	0.9830	0.9783	0.9806
D3-SACNN	0.9868	0.9859	0.9863

connections between the front and back convolutional layers. The model used 6 residual blocks with different dilation factors increased exponentially.

n-CBDC [22]: In order to improve the detection efficiency of DGA domains with obvious pronunciation and based on the vocabulary, this method used n-gram to extract the semantic relations of domains and combines them with convolutional neural networks to detect DGA domains. In the following experiment, according to the conclusion of the paper, we choose n = 2 for n-gram.

1) BINARY CLASSIFICATION

Table 6 shows the binary classification results of different algorithms on the data set. These results show that our model is better than other 3 algorithms in the detection rate of DGA domains. Our algorithm precision is 0.9868, recall value is 0.9859, and f1 value is 0.9863. All the three indicators perform the best compared to other existing methods. D3-SACNN has a relatively balanced performance in precision and recall as well, both exceeding 0.985. The other three algorithms are quite different in these two indicators, and the recall value is between 0.9783 and 0.9801.

Table 7 shows the performance of the algorithms over each DGA family in the data set. The last column is the number of DGA domains in the corresponding family. In the binary classification experiment, when a benign sample is misclassified as the DGA sample, it can not be specified as a specific DGA family. Therefore precision value can not be calculated for each DGA family. So we only display the recall values in the table. Overall, our algorithm performs the best among various DGA families. It achieves the highest values in both macro average and micro average, which are consistent with the results of the previous table 6. For each DGA family, the performance of D3-SACNN is also the best. All four algorithms can detect the domains in *gameover* and *dyre* families with recall values to be 1. The domains of these two families are all randomly distributed strings of numbers and letters, and the classification algorithms can distinguish them easily.

For the pronounceable DGA families, our algorithm has shown obvious advantages. For *suppobox*, the D3-SACNN algorithm shows a remarkable advantage, with a recall value of 0.9755, while the recall values of other algorithms is between 0.7466 and 0.8556. It is noted that the number of samples of the *suppobox* is small, and the domains of this DGA family are composed of a combination of words. For

TABLE 7. Recall score for binary classification over each family.

Domain Type	**Recall**				**Support**
	LSTM	Dilated -CNN	n-gram -CNN	SA -CNN	
benign	0.9794	0.9912	0.9794	0.9861	61408
pykspa_v1	0.9900	0.9860	0.9839	0.9906	7990
murofet	0.9960	0.9870	0.9940	0.9970	999
ramnit	0.9705	0.9470	0.9659	0.9697	3757
gameover	1.0000	1.0000	0.9991	1.0000	2352
ranbyus	0.9972	0.9944	0.9981	0.9981	2159
locky	0.9474	0.9298	0.9386	0.9298	228
simda	0.9921	0.9918	0.9755	0.9960	4285
qadars	0.9758	0.9667	0.9848	0.9848	330
symmi	0.7938	0.6639	0.7521	0.8260	839
banjori	1.0000	1.0000	0.9999	1.0000	7945
tinba	0.9949	0.9913	0.9963	0.9949	8028
rovnix	0.9999	0.9989	0.9995	0.9999	7912
dyre	1.0000	1.0000	1.0000	1.0000	195
suppobox	0.8229	0.7466	0.8556	0.9755	367
chinad	0.9951	0.9706	0.9902	1.0000	204
necurs	0.9709	0.9386	0.9607	0.9702	1579
cryptolocker	0.9950	0.9802	0.9950	0.9901	202
shiotob	0.9782	0.9596	0.9669	0.9911	1238
shifu	0.8291	0.7785	0.8418	0.8692	474
emotet	0.9975	0.9967	0.9984	0.9981	7925
virut	0.8359	0.7561	0.7876	0.8257	1968
Macro Average	0.9573	0.9352	0.9529	0.9679	122384
Micro Average	0.9812	0.9824	0.9790	0.9856	122384

these reasons this DGA family is the most difficult to classify. For *symmi*, the D3-SACNN algorithm performs best too, with a recall value of 0.8260, and the recall values of other algorithms are between 0.6639 and 0.7938. The recall values of all different algorithms against the *banjori* family reached 0.9999 or 1. This is mainly because the length of this family domain is longer, which is quite different from the length distribution of the benign domains. This difference isconducive to classification. In addition, *banjori* has a large number of samples, and the training process can better extract feature information. For *side*, the performance difference is small. Among them, the D3-SACNN algorithm performs best, with a recall value of 0.996, and the recall values of the LSTM and Dilated-CNN algorithms exceed 0.99. The overall classification performance of the D3-SACNN algorithm is the best in this experiment, and the overall classification performance of the pronounceable DGA class is better than other algorithms as well.

2) THE MULTI-CLASSIFICATION

Table 8 shows the performance results of the four algorithms in the multi-classification experiment. Like the previous experimental results, the D3-SACNN algorithm has

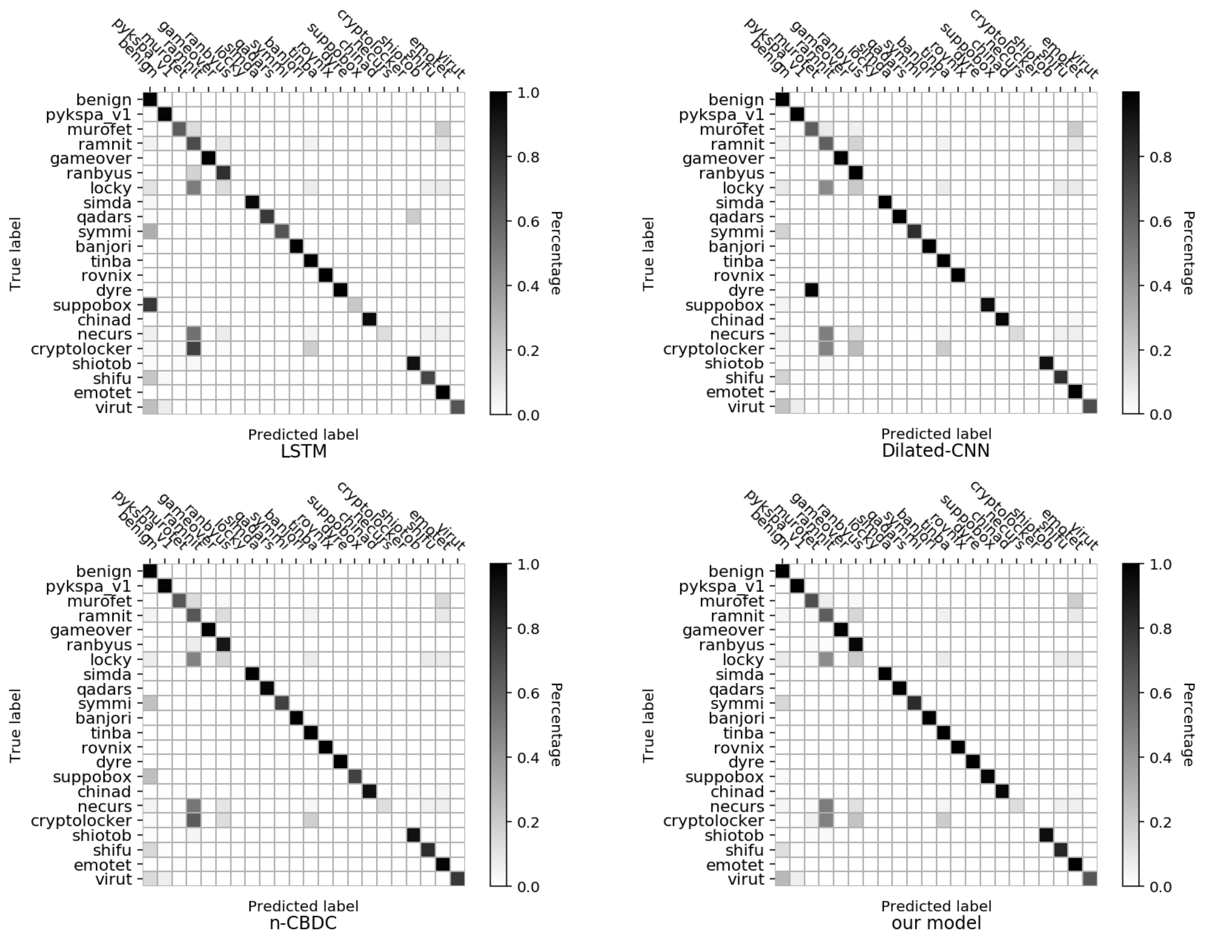

FIGURE 3. Confusion matrix of multi-classification results.

TABLE 8. Results of multi-classification.

Domain Type	Precision				Recall				F1-Score				Support
	LSTM	Dilated -CNN	n-gram -CNN	SA -CNN	LSTM	Dilated -CNN	n-gram -CNN	SA -CNN	LSTM	Dilated -CNN	n-gram -CNN	SA -CNN	
benign	0.9704	0.9814	0.9807	0.9824	0.9847	0.9875	0.9853	0.9893	0.9775	0.9845	0.9830	0.9858	61408
pykspa_v1	0.9679	0.9692	0.9705	0.9715	0.9819	0.9924	0.9834	0.9927	0.9748	0.9806	0.9769	0.9820	7990
murofet	0.8856	0.6980	0.7429	0.7744	0.6356	0.6316	0.6567	0.6907	0.7401	0.6632	0.6971	0.7302	999
ramnit	0.5809	0.6561	0.6082	0.6517	0.7006	0.6231	0.6574	0.6300	0.6351	0.6392	0.6319	0.6407	3757
gameover	0.9996	0.9996	0.9974	1.0000	0.9949	0.9953	0.9949	0.9932	0.9972	0.9974	0.9962	0.9966	2352
ranbyus	0.7674	0.6799	0.7187	0.6936	0.8161	0.9907	0.9134	0.9699	0.7910	0.8064	0.8044	0.8088	2159
locky	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	228
simda	0.9672	0.9609	0.9702	0.9769	0.9648	0.9979	0.9869	0.9951	0.9660	0.9790	0.9785	0.9859	4285
qadars	0.9585	0.9676	0.9727	0.9703	0.7697	0.9939	0.9727	0.9909	0.8538	0.9806	0.9727	0.9805	330
symmi	0.8470	0.9031	0.9122	0.9328	0.6663	0.8224	0.7426	0.8272	0.7458	0.8609	0.8187	0.8768	839
banjori	0.9980	0.9995	0.9999	0.9996	1.0000	0.9996	0.9971	1.0000	0.9990	0.9996	0.9985	0.9998	7945
tinba	0.9536	0.9603	0.9602	0.9550	0.9961	0.9963	0.9946	0.9985	0.9744	0.9779	0.9771	0.9763	8028
rovnix	0.9996	0.9985	0.9984	0.9994	0.9958	0.9961	0.9957	0.9958	0.9977	0.9973	0.9970	0.9976	7912
dyre	1.0000	0.0000	0.9898	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	0.0000	0.9949	1.0000	195
suppobox	0.9877	0.8060	0.9545	0.9253	0.2180	0.9510	0.7439	0.9782	0.3571	0.8725	0.8361	0.9510	367
chinad	0.9949	0.9900	0.9948	0.9950	0.9608	0.9706	0.9314	0.9755	0.9776	0.9802	0.9620	0.9851	204
necurs	0.9507	0.9686	0.9623	0.9333	0.1343	0.1368	0.1292	0.1330	0.2353	0.2397	0.2278	0.2328	1579
cryptolocker	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	202
shiotob	0.9190	0.9764	0.9631	0.9717	0.9346	0.9370	0.9281	0.9443	0.9267	0.9563	0.9453	0.9578	1238
shifu	0.6920	0.6944	0.6610	0.6728	0.7300	0.8101	0.8228	0.8544	0.7105	0.7478	0.7331	0.7528	474
emotet	0.9186	0.9216	0.9236	0.9212	0.9982	0.9986	0.9825	0.9991	0.9568	0.9586	0.9521	0.9586	7925
virut	0.8290	0.8826	0.7869	0.9128	0.6677	0.6951	0.7805	0.6596	0.7397	0.7777	0.7837	0.7658	1968
Macro Average	0.8267	0.7734	0.8213	0.8291	0.7341	0.7512	0.7818	0.8008	0.7526	0.7454	0.7849	0.7984	122384
Micro Average	0.9467	0.9538	0.9518	0.9564	0.9467	0.9538	0.9518	0.9564	0.9467	0.9538	0.9518	0.9564	122384

the best F1 results in macro average and micro average, which are 0.7984 and 0.9564, respectively. Let's focus on the pronounceable DGA families class. For *suppobox* with a small number of samples, the F1 value of CA-CNN has obvious advantages, and the F1 value is 0.9510. The F1 values of the other three algorithms are between 0.3571 and 0.8725. For *side*, *symmi*, CA-CNN's F1 values are all the highest, respectively 0.9859 and 0.8768. For *banjori*, the four algorithms can complete the classification task very well, and the F1 value is about 0.999. The results of multi-classification experiments show that the performance of CA-CNN is also the best, especially for pronounceable DGA families.

In order to further analyze the specific situation of the classification, we made a confusion matrix of the classification results, as shown in Figure 3. The vertical axis represents the real family labels, and the horizontal axis represents the predicted family labels. For a certain family, on the horizontal axis, the diagonal value is the correct classification ratio, and the other positions are the prediction error ratios. It can also be seen that CA-CNN classifies fewer DGA samples as *benign*, while the LSTM algorithm predicts more DGA samples as *benign*. At the same time, we noticed that from table 8, all algorithms can not classify for *locky* and *cryptolocker* DGA families correctly, and indicators are all 0. In addition, the F1 value for *necurs* is also lower than 0.24. It can be seen from the figure that *locky*, *cryptolocker* and *necurs* are mostly misclassified as *murofet* and *gameover*. We found that the character distributions of these types of DGA are very similar, and these families can be classified as a super family [14]. In addition, the number of samples in *locky* and *cryptolocker* is much smaller than that of *murofet* and *gameover*, the imbalance of the data set sizes causes families with small numbers of samples to be easily misclassified into families with similar character distributions but more samples.

3) LEAVE-CLASS-OUT BINARY CLASSIFICATION

The leave-class-out binary classification experiment is to remove one DGA family from the training set each time, and then use the DGA family domains as a test set. The remaining DGA domains and benign domains are used as the training set. Since the DGA domains removed each time do not appear in the training set, this experiment can simulate and evaluate the model's ability to detect various unknown or newly emerging DGA family domains. Because experiments and training cannot include all DGA families, and the newly emerging DGA family cannot be predicted in advance, the results of this type of experiment are very meaningful in the actual network environment. We conducted the experiments on the 21 DGA families in turn. Since there is only one DGA family data in the test set, there will be no other families of data being mistakenly classified into this family. Therefore, the experiments will focus on the metric of the recall values, that is, percentages of DGA domains are correctly classified.

The test results are shown in table 9. On the whole, D3-SACNN has the best performance with the highest micro-average and macro-average values. Specifically, different

TABLE 9. Recall score for leave-class-out binary classification.

Domain Type	**Recall**				**Support**
	LSTM	Dilated -CNN	n-gram -CNN	SA -CNN	
pykspa_v1	0.8158	0.7887	0.7714	0.8664	40000
murofet	0.9979	0.9937	0.9987	0.9994	4781
ramnit	0.9539	0.9563	0.9600	0.9786	18814
gameover	1.0000	1.0000	0.9996	1.0000	11998
ranbyus	0.9973	0.9961	0.9982	0.9980	10593
locky	0.9404	0.9396	0.9499	0.9655	1158
simda	0.3376	0.3064	0.4344	0.4647	21645
qadars	0.9078	0.9050	0.9572	0.9556	1800
symmi	0.4901	0.4904	0.6001	0.7460	4256
banjori	0.7366	0.3051	0.7225	0.9178	40000
tinba	0.9881	0.9898	0.9891	0.9980	40000
rovnix	0.9995	0.9992	0.9984	0.9997	40000
dyre	1.0000	1.0000	0.9740	1.0000	1000
suppobox	0.0092	0.0069	0.0086	0.0137	1746
chinad	1.0000	1.0000	1.0000	1.0000	1000
necurs	0.9607	0.9684	0.9665	0.9845	7918
cryptolocker	0.9950	0.9930	0.9940	0.9970	1000
shiotob	0.9507	0.8941	0.9384	0.9683	5989
shifu	0.8186	0.7308	0.7615	0.9799	2541
emotet	0.9956	0.9943	0.9977	0.9995	40000
virut	0.4728	0.2695	0.4164	0.7019	9721
Macro Average	0.8270	0.7870	0.8303	0.8826	14569
Micro Average	0.8557	0.7855	0.8549	0.9119	14569

algorithms have higher detection rates for most pseudorandom domains, such as *gameover*, *dyre* and *chinad*, which have almost 100% detection rates. For the pronounceable DGA families, D3-SACNN also showed the best performance. Taking the *banjori* DGA family as an example, the recall value of D3-SACNN reaches 0.9178, and the recall values of the other three algorithms are between 0.3 and 0.73. At the same time, we found that all models have very low recall values for *suppobox* DGA domains. The *suppobox* DGA domains are random combinations of two English words. "subjecttravel" is a *suppobox* DGA sample. Without any contextual information, we can hardly confirm that the domain is a DGA domain. Because a benign domain is usually a combination of words too. In the previous binary classification experiments, because there are samples of this family in the training set, the model can better detect these domains in the family efficiently through sufficient training.

V. CONCLUSION AND FUTURE WORK

DGA domains are used to evade detection methods and play a key role in botnets. This paper proposes a self-attention-based convolutional network D3-SACNN to detect DGA domains, and then can find botnet hosts that initiate DGA domain queries. Our method only needs the domain itself, without context information and other related information, and avoids the shortcomings of manually extracting features. The input of the D3-SACNN model is the domains to be detected, and the domain characters are vectorized through the embedding layer. The convolutional layers of 4 successively increasing convolution kernels are used to automatically extract the implicit features of the domains. In order to better obtain the long-distance dependence between features, we use a self-attention mechanism and provide different inputs for the self-attention module. Experiments based on public data sets show that this method can distinguish DGA domains with high accuracy. Especially for pronounceable DGA domains, D3-SACNN is superior to other methods in the detection indicators. In the experiment, we also observed that if there is no or only a very small number of pronounceable DGA domains in the training set, the model cannot complete the detection task very well. This is the difficulty faced by all character-based methods for detecting DGA domains. In the future, we will further study the relationship between benign domains and DGA domains in combination with domain query behavior to improve the accuracy and robustness of the model.

REFERENCES

[1] A. Zareh and H. R. Shahriari, "BotcoinTrap: Detection of bitcoin miner botnet using host based approach," in *Proc. Conf. Inf. Secur. Cryptol.*, Aug. 2018, pp. 1–6, doi: 10.1109/ISCISC.2018.8546867.

[2] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1663–1677, Oct. 2012, doi: 10.1109/TNET.2012.2184552.

[3] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *Proc. Secur. Symp.*, 2016, pp. 263–278.

[4] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2009, pp. 635–647, doi: 10.1145/1653662.1653738.

[5] R. Villamarin-Salomon and J. C. Brustoloni, "Identifying botnets using anomaly detection techniques applied to DNS traffic," in *Proc. 5th IEEE Consum. Commun. Netw. Conf.*, Jan. 2008, pp. 476–481, doi: 10.1109/ccnc08.2007.112.

[6] S. Yadav and A. L. N. Reddy, "Winning with DNS failures: Strategies for faster botnet detection," in *Proc. Secur. Privacy Commun. Netw.*, Sep. 2011, pp. 446–459, doi: 10.1007/978-3-642-31909-9_26.

[7] T. S. Wang, H.-T. Lin, W.-T. Cheng, and C.-Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Comput. Secur.*, vol. 64, pp. 1–15, Jan. 2017, doi: 10.1016/j.cose.2016.10.001.

[8] M. Antonakakis, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," in *Proc. 21st USENIX Conf. Secur. Symp.*, Berkeley, CA, USA, 2012, p. 24. Accessed: Apr. 3, 2018. [Online]. Available: http://dl.acm.org/citation.cfm?id=2362793.2362817

[9] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Proc. Int. Conf. Detection Intrusions Malware*, Jul. 2014, pp. 192–211, doi: 10.1007/978-3-319-08509-8_11.

[10] V. Tong and G. Nguyen, "A method for detecting DGA botnet based on semantic and cluster analysis," in *Proc. 7th Symp. Inf. Commun. Technol.*, Ho Chi Minh City, Viet Nam, Dec. 2016, pp. 272–277, doi: 10.1145/3011077.3011112.

[11] D.-T. Truong and G. Cheng, "Detecting domain-flux botnet based on DNS traffic features in managed network," *Secur. Commun. Netw.*, vol. 9, no. 14, pp. 2338–2347, Sep. 2016, doi: 10.1002/sec.1495.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[14] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, *arXiv:1611.00791*.

[15] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA domains with recurrent neural networks and side information," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, New York, NY, USA, 2019, pp. 1–10, doi: 10.1145/3339252.3339258.

[16] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, Jan. 2018, doi: 10.1016/j.neucom.2017.11.018.

[17] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, vol. 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015, pp. 649–657. Accessed: Mar. 18, 2019. [Online]. Available: http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

[18] H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification," in *Proc. Workshops AAAI Conf. Artif. Intell.*, Jun. 2018, pp. 29–36. Accessed: Mar. 18, 2019. [Online]. Available: https://www.aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16578

[19] J. Huang, P. Wang, T. Zang, Q. Qiang, Y. Wang, and M. Yu, "Detecting domain generation algorithms with convolutional neural language models," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Aug. 2018, pp. 1360–1367, doi: 10.1109/TrustCom/BigDataSE.2018.00188.

[20] S. Zhou, L. Lin, J. Yuan, F. Wang, Z. Ling, and J. Cui, "CNN-based DGA detection with high coverage," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2019, pp. 62–67, doi: 10.1109/ISI.2019.8823200.

[21] J. J. Koh and B. Rhodes, "Inline detection of domain generation algorithms with context-sensitive word embeddings," in *Proc. IEEE Int. Conf. Big Data*, Seattle, WA, USA, Dec. 2018, pp. 2966–2971, doi: 10.1109/BigData.2018.8622066.

[22] C. Xu, J. Shen, and X. Du, "Detection method of domain names generated by DGAs based on semantic representation and deep neural network," *Comput. Secur.*, vol. 85, pp. 77–88, Aug. 2019, doi: 10.1016/j.cose.2019.04.015.

[23] J. Lee, J. Kwon, H.-J. Shin, and H. Lee, "Tracking multiple C&C botnets by analyzing DNS traffic," in *Proc. 6th IEEE Workshop Secure Netw. Protocols*, Oct. 2010, pp. 67–72, doi: 10.1109/NPSEC.2010.5634445.

[24] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, p. 14, 2014, doi: 10.1145/2584679.

[25] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak, "Detecting DGA malware using NetFlow," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 1304–1309, doi: 10.1109/INM.2015.7140486.

[26] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: A scalable botnet detection method for large-scale DNS traffic," *Comput. Netw.*, vol. 97, pp. 48–73, Mar. 2016, doi: 10.1016/j.comnet.2015.12.008.

[27] K. Lei, Q. Fu, J. Ni, F. Wang, M. Yang, and K. Xu, "Detecting malicious domains with behavioral modeling and graph embedding," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 601–611, doi: 10.1109/ICDCS.2019.00066.

[28] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proc. 10th Annu. Conf. Internet Meas.*, New York, NY, USA, 2010, pp. 48–61, doi: 10.1145/1879141.1879148.

[29] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017, doi: 10.1109/ACCESS.2017.2779939.

[30] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 4945–4949, doi: 10.1109/ICASSP.2016.7472618.

[31] Y. Cheng, "Agreement-based joint training for bidirectional attention-based neural machine translation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, Jul. 2016, pp. 2761–2767.

[32] D. R. Radev, E. Hovy, and K. McKeown, ''Introduction to the special issue on summarization,'' *Comput. Linguistics*, vol. 28, no. 4, pp. 399–408, 2002, doi: 10.1162/089120102762671927.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ''Attention is all you need,'' 2017, *arXiv:1706.03762*.

[34] H. Du and J. Qian, ''Hierarchical gated convolutional networks with multi-head attention for text classification,'' in *Proc. 5th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2018, pp. 1170–1175, doi: 10.1109/ICSAI.2018.8599366.

[35] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' 2015, *arXiv:1512.03385*.

[36] *Netlab OpenData Project*. Accessed: Sep. 1, 2020. [Online]. Available: http://data.netlab.360.com/

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ''Dropout: A simple way to prevent neural networks from overfitting,'' *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

KEJUN ZHAO received the B.S. and M.S.degrees from Shandong University, Jinan, China, in 2002 and 2009, respectively, where he is currently pursuing the Ph.D. degree in computer science and technology. He is currently an Engineer with the Informatization Office, Shandong University. He has authored seven refereed articles in the areas of network security and data analysis. His current research interests include network security, natural language processing, and deep learning.

WEI GUO received the master's degree from the Department of Computer Science and Technology, Shandong University, in 2005, and the Ph.D. degree in engineering from Shandong University, in 2015. He is currently an Engineer and a Master's Supervisor. He has participated in more than ten national, provincial, and ministerial level projects. He has published more than 20 articles in important journals or conferences and an academic monograph. There are more than ten patents for invention in the application field, of which seven are first invention applicants. His current research interests include big data technology and intelligent data analysis. He is currently with the committee of CCF TCSC and CCF TCCC. At the same time, he was a recipient of several prizes for Scientific and Technology Progress (STP), including one Second Class Prize for STP of State Education Ministry of China, one First Class Prize, and two Second Class Prizes for STP of Shandong Province. He was also a recipient of one First Class Prize for Ministry of Education Teaching Achievement Award and one Second Class Prize for Teaching Achievement Award of Shandong Province.

FENGLIN QIN received the Ph.D. degree in communication engineering from Shandong University, Jinan, China, in 2011. He is currently a Senior Engineer at Informatization Office, Shandong University. His current research interests include data mining, network measurement, and network security.

XINJUN WANG received the Ph.D. degree in computer science from Shandong University, in 2004. He is currently a Professor and a Ph.D. Supervisor. He is also an Academic Leader of the Outstanding Innovation Team. He has received nearly 20 research grants at national, provincial, and ministerial levels. He has published more than 40 high-level academic papers in journals or international conferences. His current research interests include big data technology and intelligent data analysis.