Hindawi
Security and Communication Networks
Volume 2021, Article ID 5518528, 12 pages
https://doi.org/10.1155/2021/5518528

Research Article

A Malicious URL Detection Model Based on Convolutional Neural Network

Zhiqiang Wang,[1,2,3] Xiaorui Ren,[1] Shuhao Li,[1] Bingyan Wang,[1] Jianyi Zhang,[1] and Tao Yang[3]

[1]*Beijing Electronic Science and Technology Institute, Beijing 100071, China*
[2]*State Information Center, Beijing 100032, China*
[3]*Key Lab of Information Network Security, Ministry of Public Security, Shanghai 200000, China*

Correspondence should be addressed to Jianyi Zhang; nese@163.com

Received 18 January 2021; Accepted 30 April 2021; Published 15 May 2021

Academic Editor: Hao Peng

With the development of Internet technology, network security is under diverse threats. In particular, attackers can spread malicious uniform resource locators (URL) to carry out attacks such as phishing and spam. The research on malicious URL detection is significant for defending against these attacks. However, there are still some problems in the current research. For instance, malicious features cannot be extracted efficiently. Some existing detection methods are easy to evade by attackers. We design a malicious URL detection model based on a dynamic convolutional neural network (DCNN) to solve these problems. A new folding layer is added to the original multilayer convolution network. It replaces the pooling layer with the k-max-pooling layer. In the dynamic convolution algorithm, the width of feature mapping in the middle layer depends on the vector input dimension. Moreover, the pooling layer parameters are dynamically adjusted according to the length of the URL input and the depth of the current convolution layer, which is beneficial to extracting more in-depth features in a wider range. In this paper, we propose a new embedding method in which word embedding based on character embedding is leveraged to learn the vector representation of a URL. Meanwhile, we conduct two groups of comparative experiments. First, we conduct three contrast experiments, which adopt the same network structure and different embedding methods. The results prove that word embedding based on character embedding can achieve higher accuracy. We then conduct the other three experiences, which use the same embedding method proposed in this paper and use different network structures to determine which network is most suitable for our model. We verify that the model designed in this paper has the highest accuracy (98%) in detecting malicious URL through these experiences.

1. Introduction

Hackers often use spam and phishing [1, 2] to trick users into clicking malicious URL, the Trojans will be implanted into the victims' computers, or the victims' sensitive information will be leaked. The technology of malicious URL detection can help users identify malicious URL and prevent users from being attacked by malicious URL. Traditionally, research on malicious URL detection adopts blacklist-based methods to detect malicious URL. This method has some unique advantages. It has high speed, has low false-positive rate, and is easy to realize. However, nowadays, the domain generation algorithm (DGA) can generate tens of thousands of different malicious domain names every day, which cannot be detected effectively by the traditional blacklist-based methods.

Researchers have been using a machine learning technique to identify malicious URL. However, these methods often need to extract the features manually, and attackers can design these features to avoid being identified. Faced with today's complex network environment, designing a more effective malicious URL detection model becomes a research focus.

This paper proposes a malicious URL detection model based on a DCNN. It adopts word embedding based on the character embedding method to extract features automatically and learn the URL's expression. Meanwhile, we verify the validity of the model through a series of contrast experiments.

In this study, our innovations and contributions are as follows:

(1) This paper proposes a malicious URL detection model based on a DCNN. The dynamic convolution algorithm adds a new folding layer to the original multilayer convolution structure. It replaces the pooling layer with the k-max-pooling layer. In the dynamic convolution algorithm, the width of feature mapping in the middle layer depends on the vector input dimension. Moreover, the pooling layer parameters are dynamically adjusted according to the length of the URL input and the depth of the current convolution layer, which helps extract more in-depth features in a wider range.

(2) In the stage of feature extraction and representation, the features are extracted from the URL sequence. The extracted features are integrated into a vector, and the vector is processed directly by the convolutional neural network to learn the classification model. This method not only simplifies the process of feature extraction, it does not depend on extracting features manually, but also combines the advantages of character embedding and word embedding. Word embedding can obtain word sequence information, which cannot be obtained by character embedding. Character embedding can process special characters and unfamiliar words in the URL. The dictionary and vector dimension are also not too big. The combination can save memory space and express the URL more effectively, which will help extract information from the URL.

(3) To prove the feasibility of the model proposed in this paper, we did a lot of comparative experiments. As for the embedding method, we conduct three contrast experiments to verify that word embedding based on character embedding achieves higher accuracy than word embedding and character embedding. We also perform three contrast experiments and prove that leveraging the network structure consisting of a DCNN and different fields extracted from the URL can achieve a better effect.

The rest of this paper is organized as follows. In Section 2, we introduce the research status of malicious URL detection methods. In Section 3, we present the malicious URL detection model and its main modules. In Section 4, we conduct experiments on the malicious URL detection model and test the embedding methods. Finally, we offer a brief discussion in Section 5.

1.1. Related Work. At present, the methods [3–5] of detecting malicious URL can be roughly divided into traditional detection methods based on blacklist and detection methods based on machine learning. Literatures [6, 7] introduce the detection method based on a blacklist. Although this method is simple and efficient, it cannot detect the newly generated malicious URL, which has severe limitations. Literature [8] points out that attackers can generate various malicious domain names through a random seed to effectively evade the traditional detection method based on a blacklist.

In literatures [9–11], researchers have applied machine learning technology to detect malicious URL. Machine learning learns the prediction model based on statistical properties and classifies a URL as a malicious URL or a benign URL. This method attempts to analyze URL and their relevant websites or web page information to extract the features. The features extracted by this method are often divided into two types, static features and dynamic features. Literature [12] obtains lexical information in URL strings, information about hosts, and sometimes HTML and JavaScript content. Literature [13] extracts a series of network traffic-related features from URL, and the support vector machine (SVM) is adopted for detection. Literature [14] proposes three methods of feature processing to optimize the classification effect. While the above methods have shown good performance, there are still some limitations. Traditional detection methods based on machine learning often require extract features manually [15, 16]. Attackers can avoid being detected by these detection methods by designing these features, making it very difficult to maintain the detection system based on traditional machine learning. Additionally, in large-scale malicious URL detection, a trained model may lose some useful information from URL.

Referring to the idea of text classification [17, 18], many researchers have proposed a variety of methods based on deep learning [19] models to detect malicious URL and judge whether a URL is malicious only according to the strings contained in the URL. These methods can automatically extract valid information in the URL. For example, literature [20] uses the cyclic neural network model at the character level to classify URL generated by DGA. Literature [21] proposes the method of extreme machine learning to detect malicious URL. Combining n-gram model with deep learning, literature [22] takes the advantages of character-level semantic features to detect whether DGA generates the URL. A variety of deep learning architectures for malicious URL detection are listed in the literature [23], including the structure of single-layer long short term memory(LSTM) [24], the structure of bidirectional LSTM [25], the combined structure of CNN and LSTM [26, 27], and the deep convolution structure [28]. On this basis, literature [29] designs a keyword-based malicious URL detection model, which combines word embedding and GRU model. Literature [30] analyzes the structures and features of different URL, extracts more features, and proposes a semisupervised training model for URL's multiclassification. Literature [31] extracts URL domain name features and instantaneous features of redirection attacks and optimizes the neural network structure to improve detection accuracy. In recent years, it has become a new research direction to detect malicious URL directly. Literature [32] takes the original URL as the input of malicious URL detection system, transforms URL into feature vectors by character embedding technology, and then uses the convolutional neural network for training, which significantly reduces the dimension of data and the amount of computation. Additionally, it can help achieve a good classification effect. In literature [30], word embedding is used to transform URL in each message into vector-matrix, which is then inputted into the convolutional neural network for analysis. Literature [33] improves the detection algorithm and adds a convolution branch on the CNN structure to extract in-depth character-level features. The disadvantage of this method is that it adopts character embedding or word embedding alone, and it is difficult to extract features in both characters and phrases. Literatures [34, 35] adopt a parallel convolutional neural network to detect malicious URL. They combine character embedding and word embedding, improve the word embedding method in the vector embedding stage, and extract the URL's character and phrase features, which improve the detection effect. However, one of their disadvantages is that the fixed CNN structure is used to detect URL. The model parameters cannot be adjusted according to the input vector's dimension, so it is difficult to extract the in-depth features in a wide range.

Based on DCNN, this paper designs a malicious URL detection model to solve the abovementioned problem. We will detail this in the following chapters. The materials and methods section should contain sufficient detail so that all procedures can be repeated. It may be divided into headed subsections if several methods are described.

2. Malicious URL Detection Model

Our paper proposes a malicious URL detection model based on convolutional neural networks. The construction of the model is shown in Figure 1. The model mainly includes three modules: vector embedding module, dynamic convolution module, and block extraction module. In the following, we will introduce each module and the detection process in detail.

3. Vector Embedding Module

In our model, a URL is inputted into the embedding layer, and we use word embedding based on character embedding to transform the URL into a vector expression. Moreover, the URL will be input into the DCNN for feature extraction.

The vector embedding module represents the input URL sequence as a suitable vector to facilitate the subsequent process. In the beginning, URL vector representation is initialized randomly. It is then inputted into the embedding layer used in the subsequent training and the most appropriate URL vector expression is obtained during the training process. This module uses an advanced word embedding method based on character embedding. The module extracts the phase information from the URL and the character-level information from the word. The information extracted will be used in subsequent training to obtain the most appropriate vector expression of the URL, and then the vector expression is inputted into the subsequent convolutional layer.

An example of a word embedding method based on character embedding is shown in Figure 2, where k represents the embedding dimension of characters and words, L_2 represents the number of words contained in a URL string, and L_3 represents the number of characters contained in each word in the URL string. In the example, we convert each URL to an id sequence of word and character, respectively. Then, the embedding layer obtains word embedding matrix EM_w and character embedding matrix EM_c during the subsequent training. The word ID sequence uses the word embedding matrix EM_w to obtain the matrix expression URL_w at the word level. The character ID sequence uses the character embedding matrix EM_c to obtain multiple word matrix expressions based on character embedding. The multiple word matrices will be merged and compressed into a matrix representation URL_{cw} of the URL at the word level. The matrix representation URL_w and the matrix representation URL_{cw} are added. We will get the final representation of the URL.

3.1. *Dynamic Convolution Module.*

The dynamic convolution module is to extract features automatically from the input data. The processing procedure of data includes 1D convolution, folding, and dynamic pooling. The DCNN can adjust parameters and extract features in a wider range based on the input length and the current convolutional layer.

When the DCNN is training, the network's upper layer's output is inputted into the network's next layer. The URL is inputted into the input layer, and it is converted to a suitable vector expression in the embedding layer. Then, the first convolution layer starts to extract features. After the data are outputted from the convolutional layer, the data tensor dimension is compressed by the folding layer and then inputted to the pooling layer for dynamic pooling. After several rounds of convolution-folding-pooling, the data are finally inputted into the fully connected layer for training, and the result is finally outputted from the output layer.

3.2. *Block Extraction Module.*

The block extraction module extracts different fields such as subdomain name, domain name, and domain name suffix from URL and encodes them as the second data branch of the detection model. In the embedding layer, the URL is converted to an appropriate vector. After passing through the embedding layer, the second data branch is merged with the first data branch, and the merged result is inputted to the fully connected layer for training. When the block extraction module extracts different fields, it can separate the top-level domain name or national domain name from the URL string.

The block extraction module can distinguish between generic top-level domains and national top-level domains. Therefore, the model can make full use of essential

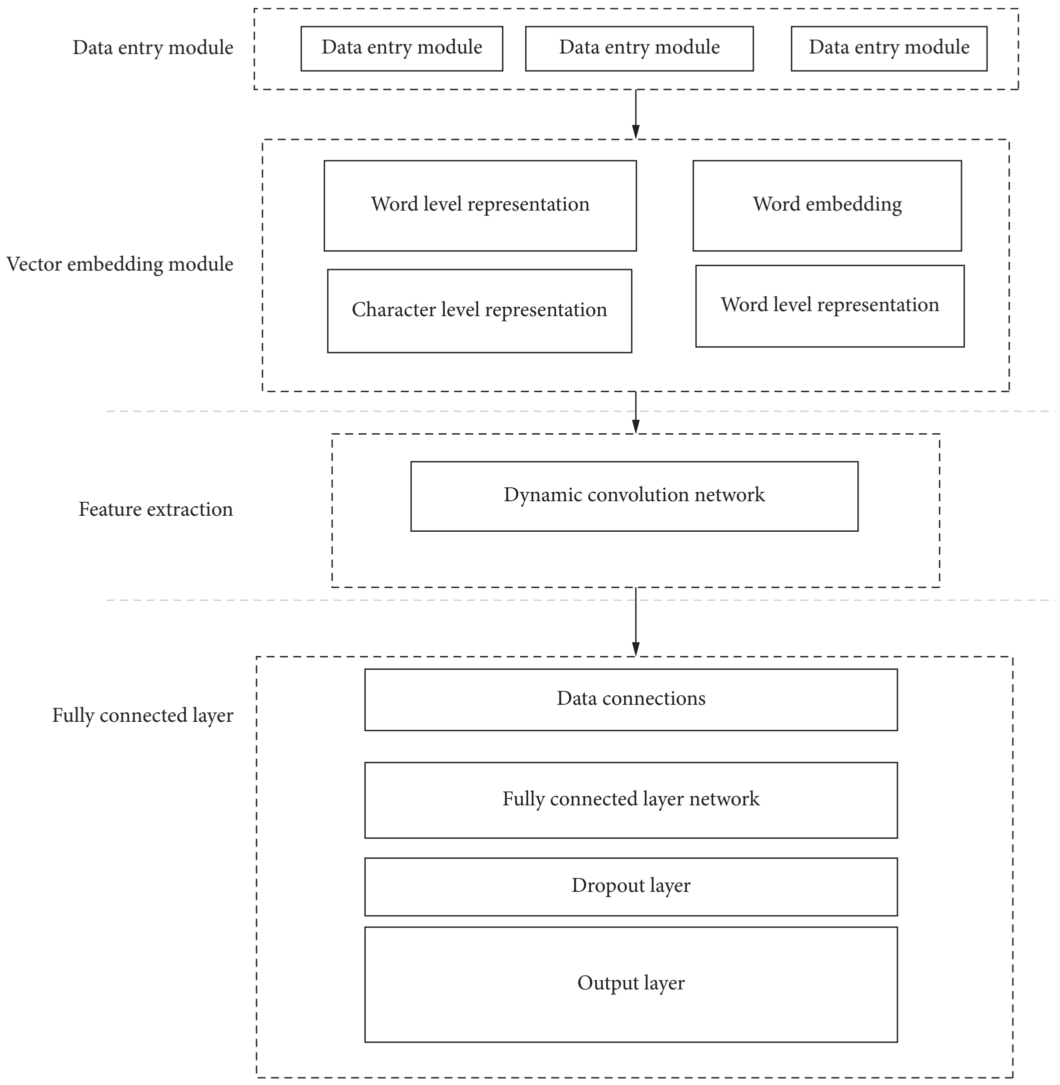

Figure 1: The detection model.

information. It takes the different fields in the URL as different features, which enriches the fully connected layer's input.

3.3. Detection Process. The detection process is as follows. First, domain name, subdomain name, and domain name suffix are sequentially extracted from URL. In the first branch of the detection model, we pad each URL to a fixed length, of which every word is marked with a specific number. The entire URL is represented as a sequence of numbers. Then, the sequences are inputted to the embedding layer and trained together with other layers. These sequences will learn the appropriate vector expression during the training process. The data stream outputted from the embedding layer is subsequently inputted into a DCNN. The output passes through a convolution layer, a folding layer, and a pooling layer in two successive rounds. In the flatten layer, the data stream is flattened. It then waits for connecting with data from the other branch, where domain name, subdomain name, and domain name suffix are marked. The different main domain name, subdomain name, or domain name suffix in each field are encoded as an independent expression. Then, the marked data are inputted into the three newly added embedding layers and obtain the appropriate vector expression. After that, the information is transformed into a suitable shape in the reshape layer and merged with the first branch's data. The two branches' outputs are combined and jointly inputted to the fully connected layer for training. We use the DCNN to extract features automatically and use different fields in URL as different features to detect malicious URL jointly. After the dropout layer, the result is outputted into the output layer.

This model can fully obtain the information carried by the different fields in the URL string and enrich the input of the fully connected layer, which improves the detection effect.

In summary, the branch of processing data is added for expanding the input of the detection model. When training in the fully connected layer, the features are extracted

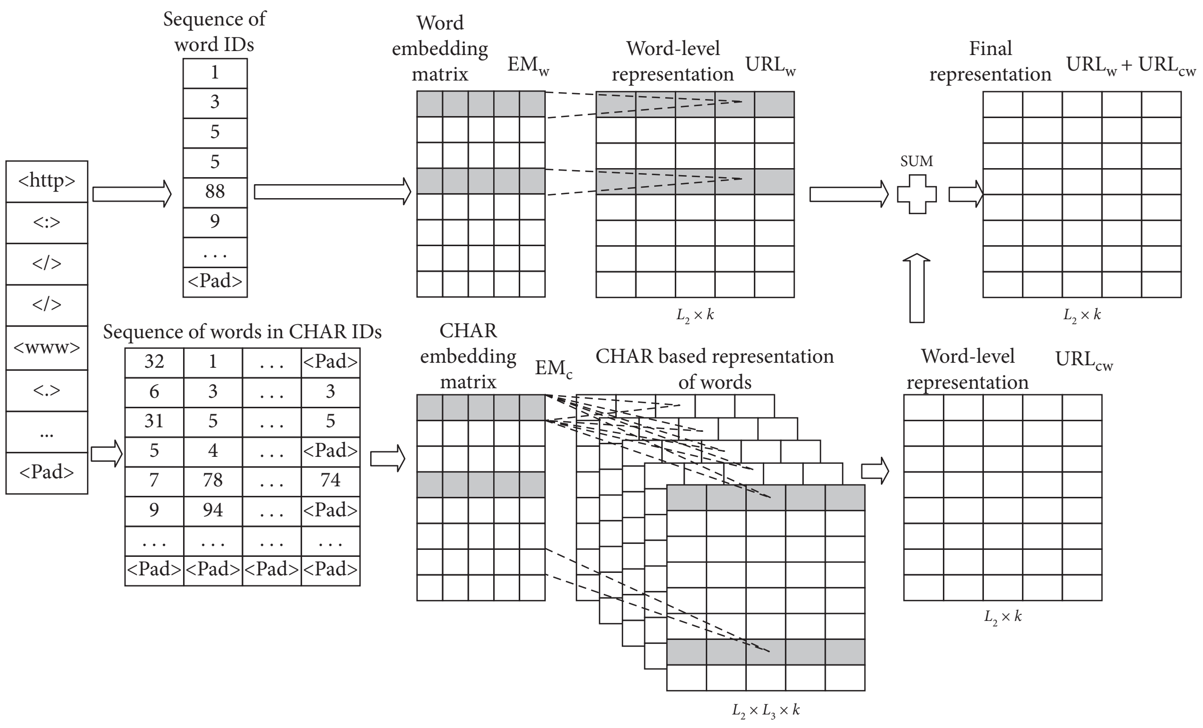

Figure 2: Example of a word embedding method based on character embedding.

automatically by the convolutional neural network and extracted artificially from the URL field. The detection model can effectively utilize critical information in the URL, such as top-level domain names and national domain names, to achieve higher accuracy and recall. Accuracy is vital, especially for detecting models, because if the accuracy is low, normal web pages may be classified as malicious websites and will be blocked.

4. Experiment and Evaluation

4.1. Experimental Environment. The experimental environment is based on the Windows operating system. The processor is i5-7500, the memory is 8 GB, the programming language is Python 3.6, and the deep learning framework is TensorFlow.

4.2. Comparative Experiments between Different Embedding Methods. Our paper adopts the word embedding method based on character embedding, which considered the advantages and disadvantages of word embedding and character embedding. The advantages of word embedding based on character embedding are as follows:

(1) This method can effectively express rare words. It takes full use of character-level and word-level information. Therefore, this method can accurately represent rare words in URL.

(2) This method can reduce the scale of the embedded matrix and reduce memory space. Meanwhile, it helps to convert a URL to a more accurate expression.

(3) This method can convert new words that do not exist in training set into more accurate vectors, thereby extracting character information.

Based on different embedding methods, we conducted three comparative experiments. Experiment 1 adopted character embedding. Experiment 2 utilized word embedding as an embedding method. Experiment 3 used word embedding based on character embedding as an embedding method. These experiments used malicious DGA URL as the training set, and the network structure was stacked CNN. We measured the accuracy, F1-score, precision, and recall ratio to evaluate the results.

Attackers can communicate with the control center through a malicious DGA domain name. A malicious DGA domain name can be treated as a string during detection. However, compared with the real malicious URL, the malicious DGA domain name contains fewer characters.

The experimental result is shown in Table 1. We find that word embedding based on character embedding achieves the highest accuracy among the above two embedding methods through these experiments. The accuracy of word embedding based on character embedding is 0.958; the accuracy of character embedding and word embedding are only 0.923 and 0.954, respectively. The recall of word embedding based on character embedding achieves 0.976, which is higher than character embedding and word embedding. We also draw ROC curves and AUC curves among the three experiments. It is shown in Figure 3.

Table 1: Experiment results.

No.	Embedding method	Network structure	Accuracy	F1-score	Recall	Precision
1	Character embedding	Stacked CNN	0.923	0.926	0.964	0.890
2	Word embedding	Stacked CNN	0.954	0.953	0.931	0.977
3	Word embedding based on character embedding	Stacked CNN	**0.958**	**0.959**	**0.976**	**0.942**

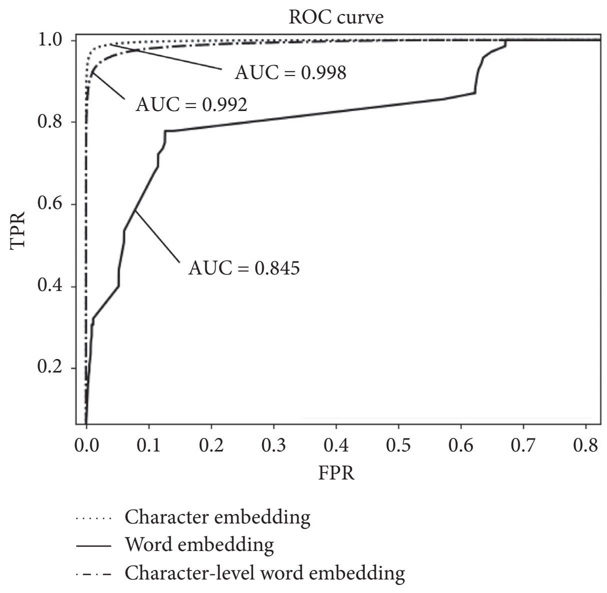

Figure 3: Comparison of ROC curves and AUC values.

4.2.1. Network Structure Experiment

(i) Experimental Dataset. We collect a large amount of URL data from github.com, uci.edu, and kaggle.com to verify the model's validity and feasibility. Besides, we select the top 1 million domain names, published on the Alexa website on May 17, 2019, as standard URL. Alexa ranks URL in terms of website traffic, including the site's top-level domains and subdomains. The dataset division is shown in Table 2; the distributions of the dataset are shown in Figures 4–6.

(ii) Experiment Setting. We designed three comparative experiments. We tried to use different network structures to determine the best solution for our model. Experiment 1 utilized the stacked CNN. Experiment 2 only leverages DCNN. Experiment 3 adopted DCNN, and it extracted different fields from the URL to participate in training. We set the same experimental parameters for these three experiments. Besides, each URL's length was set to 200 words, and the vector embedding dimension was 32. The DCNN included two convolutional layers. The number of convolution kernels was set to 128. The DCNN was finally trained by one fully connected layer, and it adopted the Adam optimization algorithm. The learning rate was 0.001, and the drop rate of the dropout layer was 0.5. In the process of the experiment, we adopted batch training, and each batch contains 100 URL.

Table 2: Dataset.

	Training set	Validation set	Test set
Malicious URL	200 k	25 k	25 k
Normal URL	200 k	25 k	25 k
Total	400 k	50 k	50 k

(iii) Experimental Results and Analysis. We measured the accuracy, F1-score, precision, and recall ratio to evaluate the results. The final results of the detection model in this paper are shown in Table 3. The accuracy reaches 0.987, the precision reaches 0.993, the F1-score reaches 0.987, and the recall ratio is 0.981. The accuracy and loss are shown in Figures 7 and 8. As the number of iterations increases, the training accuracy increases continuously, and the fitting degree of the model is ideal. At the same time, the loss continues to decrease.

We also list the experimental results of other comparative experiments, as shown in Table 4; the network structure DCNN + extracting fields obtained a better effect than the other two network structures. The accuracy reaches 0.987, and the precision is 0.993. The ROC curves are drawn, and

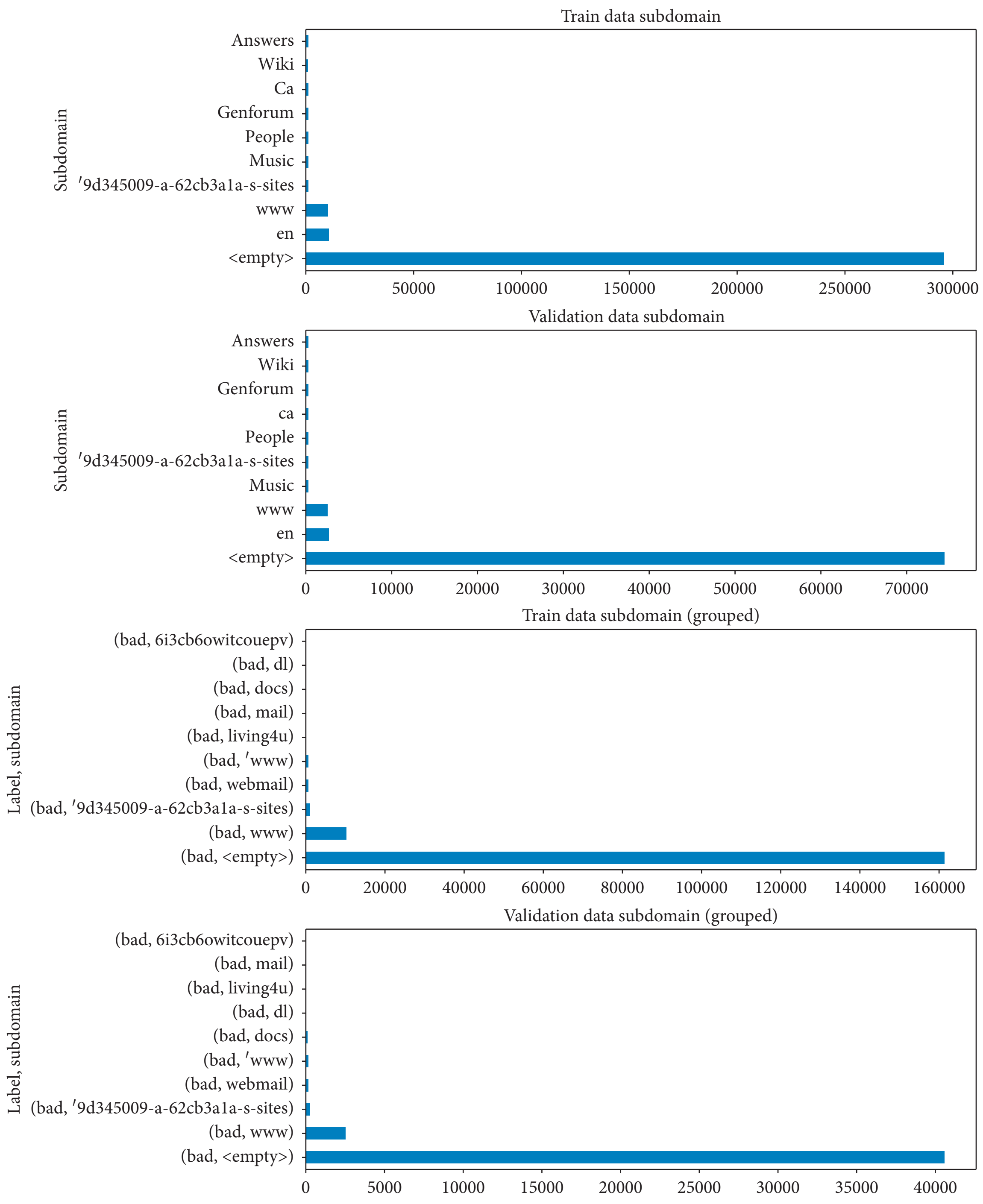

FIGURE 4: Distribution of data subdomains.

the AUC value is calculated to facilitate comparison and analysis. They are shown in Figure 9. We can see the TPR of DCNN + extraction is the first to reach 0.9993.

From the results, it can be seen that the best detection effect is obtained in Experiment 3. The high accuracy indicates that benign samples are less likely to be misjudged as malicious samples. Experiments show that the URL can be adequately expressed, critical features can be extracted to obtain better detection results using DCNN and the word embedding based on character embedding. Extracting different fields from the URL can make full use of keywords in the URL to improve detection accuracy and precision. The abovementioned experiments verify the validity of the detection model in this paper.

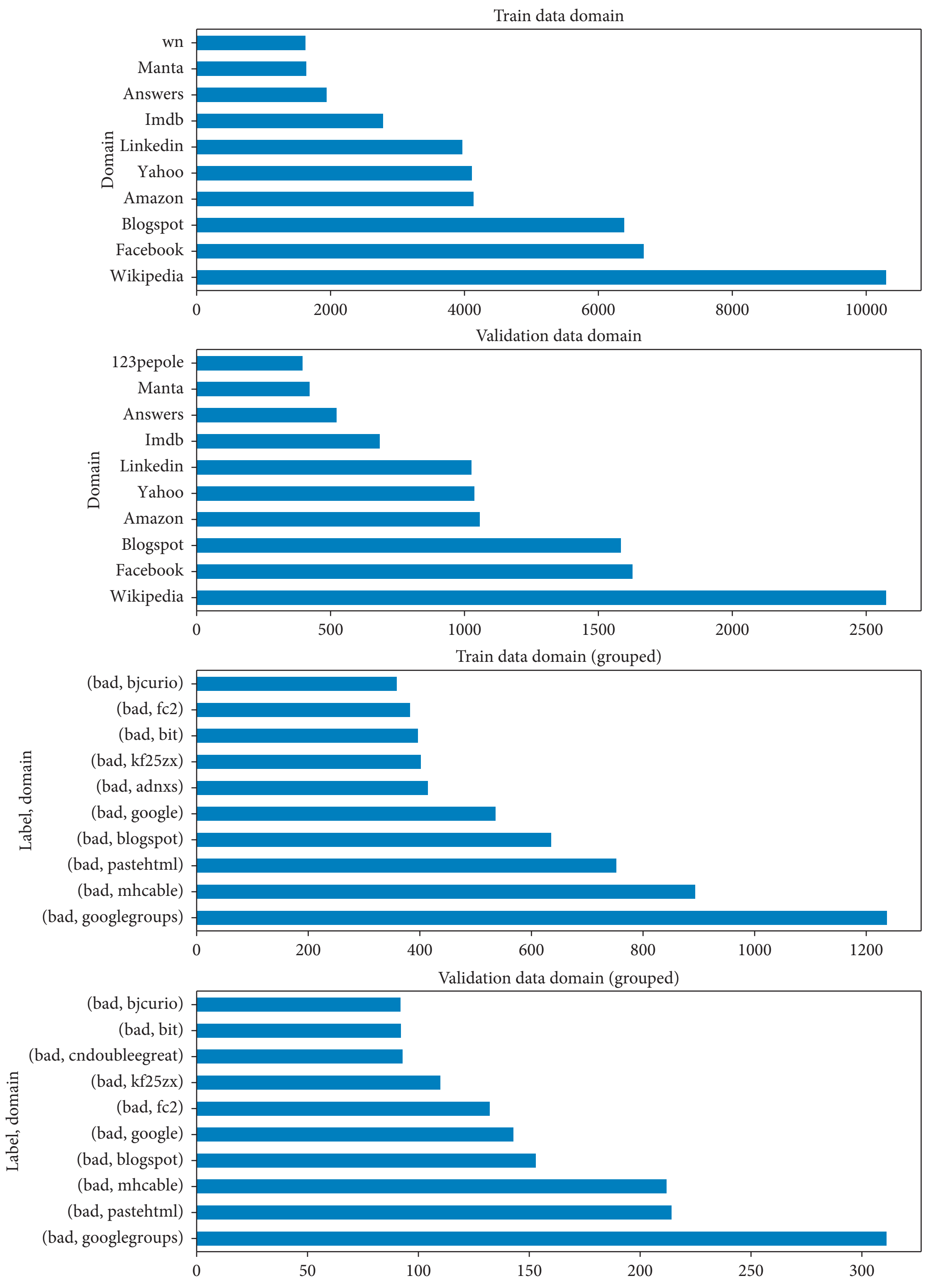

Figure 5: Distribution of data domain.

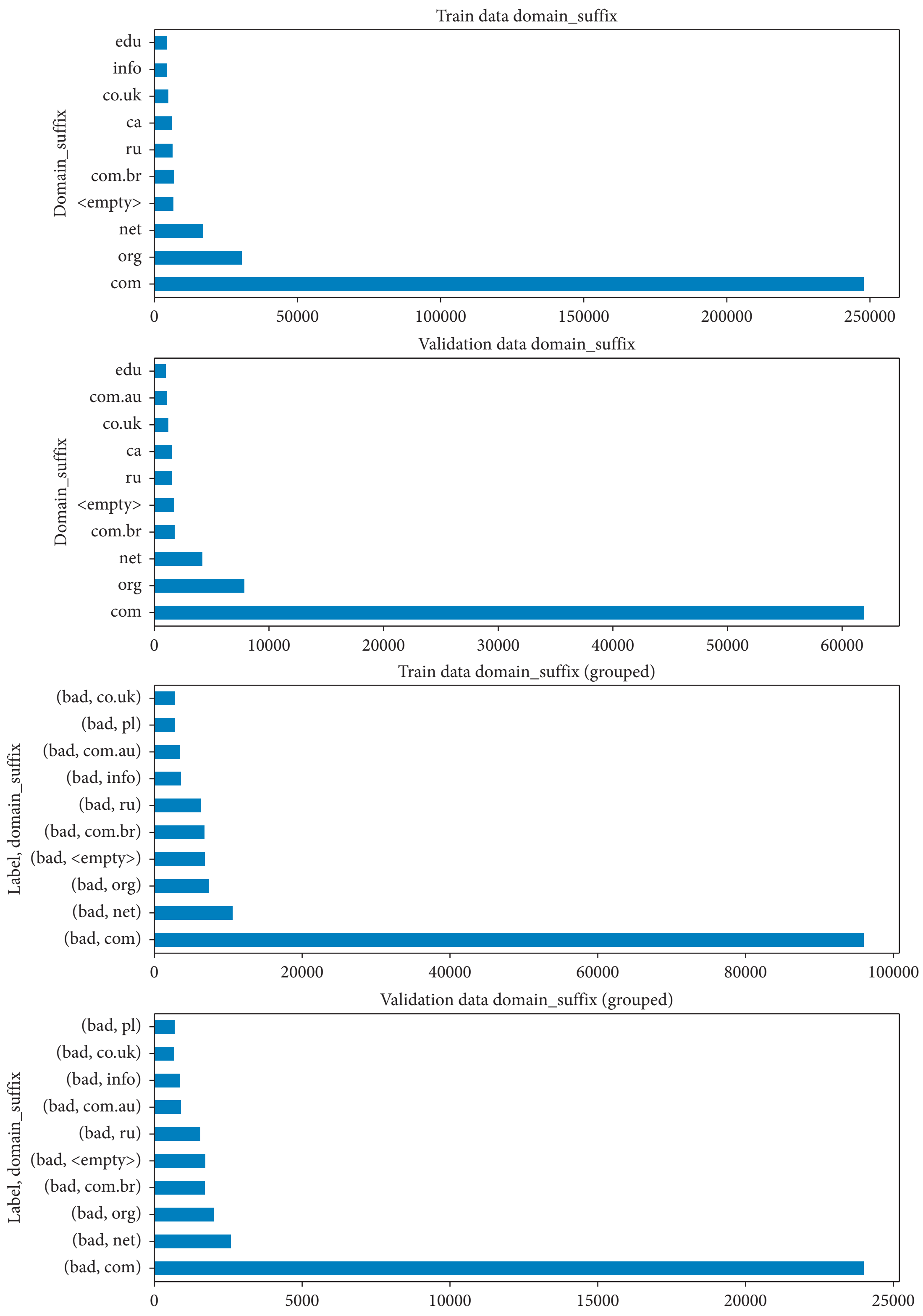

FIGURE 6: Distribution of data domain_suffix.

TABLE 3: Experimental results.

Detection indicator	Meaning	Value
Accuracy	((TP + TN)/P + N)	0.987
F1-score	(2TP/(2TP + FP + FN))	0.987
Recall	(TP/(TP + FN))	0.981
Precision	(TP/(TP + FP))	0.993

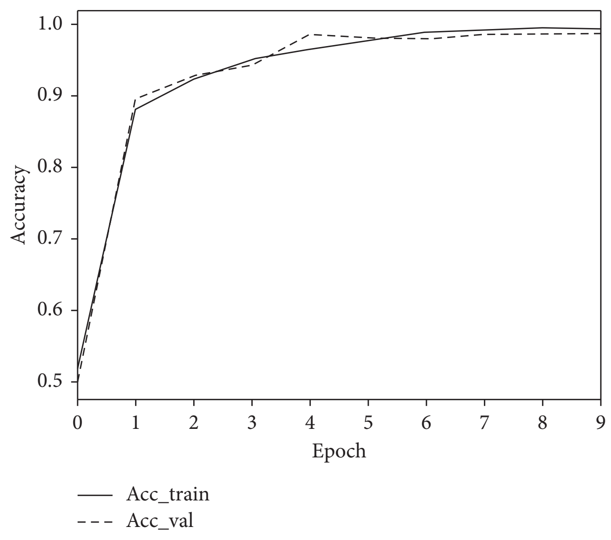

Figure 7: Training accuracy and verification accuracy.

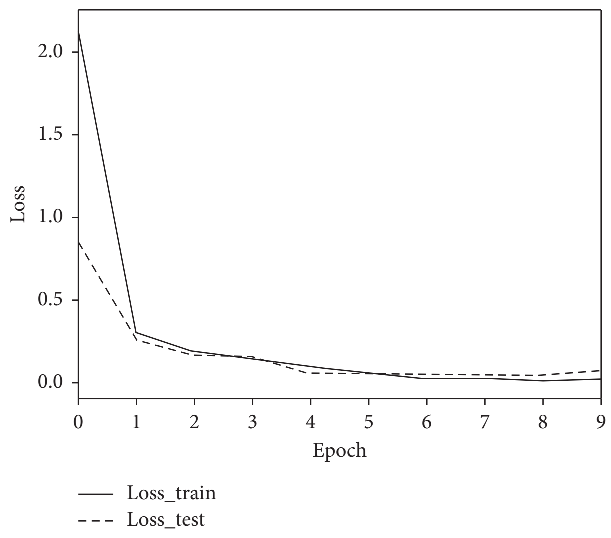

Figure 8: Training loss and validation loss.

Table 4: Experiment results.

No.	Embedding method	Network structure	Accuracy	F1-score	Recall	Precision
1	Word embedding based on character embedding	Stacked CNN	0.958	0.959	0.976	0.942
2	Word embedding based on character embedding	DCNN	0.961	0.960	0.936	0.984
3	Word embedding based on character embedding	DCNN + extracting fields	**0.987**	**0.987**	**0.981**	**0.993**

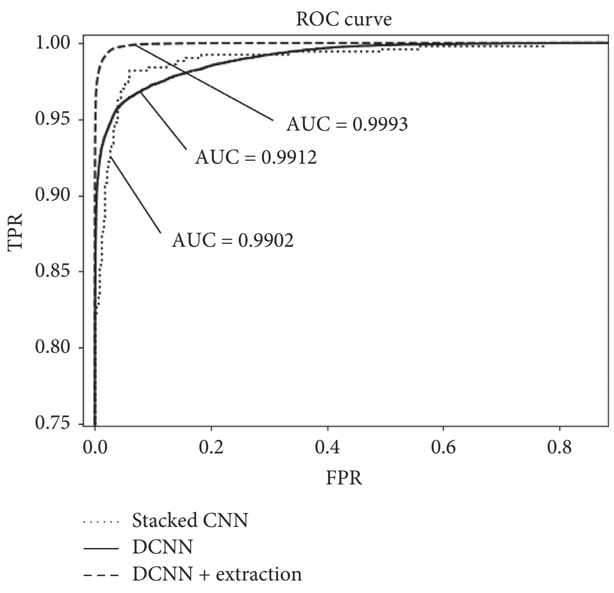

Figure 9: Comparison of ROC curves and AUC values in three experiments.

5. Conclusions

This paper aims to design a new malicious URL detection model based on deep learning. We designed a word embedding method based on character embedding, and the vector expression of a URL is learned automatically by combining character embedding with word embedding. DCNN for malicious URL detection is designed. According to the length of the input vector and the depth of the current convolution layer, the pooling layer parameters are dynamically adjusted to extract features in a wider range. We coordinated the relationship between different modules and adjusted network parameters. Besides, the real URL and malicious DGA URL in the real network are collected, and a series of experiments are designed and conducted. The results and various indicators are compared and analyzed to demonstrate the validity of the detection model.

The detection model achieves the expected effect in experiments. However, considering that the network traffic in the test environment and the real network are different, and with the development of the Internet, types of malicious URL are more diverse. It is necessary to timely update the model in the actual scenario. Therefore, to better adapt to the requirements of various complex application scenarios, we plan to study how to simplify the detection model's architecture and shorten the training time while keeping the detection performance unchanged in the future.

Data Availability

The URL data used to support the findings of this study have been deposited in the malicious and benign URL dataset: https://gitee.com/blackwall/UrlDetect/blob/master/urldetect/%E5%AE%9E%E9%AA%8C%E4%BB%A3%E7%A0%81.zip.

Conflicts of Interest

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

Acknowledgments

The authors gratefully acknowledge the Key Laboratory of Network Assessment Technology of the Institute of Information Engineering. This research was financially supported by the Key Lab of Information Network Security, Ministry of Public Security (C19614), China Postdoctoral Science Foundation (2019M650606), and First-Class Discipline Construction Project of Beijing Electronic Science and Technology Institute (3201012).

References

[1] D. J. Lemay, R. B. Basnet, and T. Doleck, "Examining the relationship between threat and coping appraisal in phishing detection among college students," *Journal of Internet Services and Information Security*, vol. 10, no. 1, pp. 38–49, 2020.

[2] H. Kim, "5G core network security issues and attack classification from network protocol perspective," *Journal of Internet Services and Information Security*, vol. 10, no. 2, pp. 1–15, 2020.

[3] K. Aram and J. O. SoK, "A systematic review of insider threat detection," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 10, no. 4, pp. 46–67, 2019.

[4] R. B. Basnet and R. Shash, "Towards detecting and classifying network intrusion traffic using deep learning frameworks," *Journal of Internet Services and Information Security*, vol. 9, no. 4, pp. 1–17, 2019.

[5] F. Valenza and M. Cheminod, "An optimized firewall anomaly resolution," *Journal of Internet Services and Information Security*, vol. 10, pp. 22–37, 2020.

[6] D. R. Patil and J. B. Patil, "Survey on malicious web pages detection techniques," *International Journal of U- and E-Service, Science and Technology*, vol. 8, no. 5, pp. 195–206, 2015.

[7] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, Rio de Janeiro, Brazil, July 2018.

[8] C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. Nascimento, and M. De Cock, "Algorithmically generated domain detection and malware family classification," in *Proceedings of the International Symposium on Security in Computing and Communication*, pp. 640–655, Springer, Singapore, September 2018.

[9] S. Garera, N. Provos, and M. Chew, "A framework for detection and measurement of phishing attacks," in *Proceedings of the ACM Workshop on Recurring Malcode*, ACM, New York, NY, USA, November 2007.

[10] D. K. M. M. Gupta, "Behind phishing: an examination of phisher modi operandi," in *Proceedings of the Usenix Workshop on Large-scale Exploits & Emergent Threats*, DBLP, San Francisco, CA, USA, April 2008.

[11] J. Ma, L. K. Saul, and S. Savage, "Beyond blacklists: learning to detect malicious Web sites from suspicious URL," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Paris, France, July 2009.

[12] H. Choi, B. B. Zhu, and H. Lee, "Detecting malicious web links and identifying their attack types," in *Proceedings of the 2nd USENIX conference on Web application development*, Boston, MA, USA, June 2011.

[13] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *Proceedings of the USENIX Security Symposium*, pp. 807–822, Austin, TX, USA, August 2016.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The People's Posts and Telecommunications Press, Beijing, China, 2016.

[15] Z. Feng, C. Shuo, and W. Xiaochuan, "Classification for DGA-based malicious domain names with deep learning architectures," in *Proceedings of the Second International Conference on Applied Mathematics and information technology*, p. 5, Shanghai, China, October 2017.

[16] C. Xu, J. Shen, and X. Du, "Detection method of domain names generated by DGAs based on semantic representation and deep neural network," *Computers & Security*, vol. 85, pp. 77–88, 2019.

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

[18] X. Zhang, J. Zhao, and Y. Lecun, "Character-level convolutional networks for text classification," 2015, https://arxiv.org/abs/1509.01626.

[19] J. Clayton and K. Bishal, "Towards detecting and classifying malicious url using deep learning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 4, pp. 31–48, 2020.

[20] Z. U. O. Wen, *Research and Design of Malicious URL Detection Algorithm on Deep Learning*, Beijing University of Posts and Telecommunications, Beijing, China, 2019.

[21] J. Yang, P. Yang, X. Jin, and Q. Ma, "Multi-classification for malicious URL based on improved semi-supervised algorithm," in *Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 143–150, IEEE, Guangzhou, China, July 2017.

[22] T. Shibahara, K. Yamanishi, and Y. Takata, "Malicious URL sequence detection using event de-noising convolutional neural network," in *Proceeings of the Communications (ICC), 2017 IEEE International Conference*, pp. 1–7, IEEE, Paris, France, May 2017.

[23] J. Woodbridge, H. S. Anderson, and A. Ahuja, "Predicting domain generation algorithms with long short-term memory networks," 2016, https://arxiv.org/abs/1611.00791.

[24] B. Dhingra, Z. Zhou, and D. Fitzpatrick, "Tweet2Vec: character-based distributed representations for social media," 2016, https://arxiv.org/abs/1605.03481.

[25] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "Tweet2vec: learning tweet embeddings using character-level CNN-LSTM encoder-decoder," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 1041–1044, ACM, Pisa, Italy, July 2016.

[26] Y. Liu, K. J. Zhao, S. Ge lian, and Y. Liu, "A fast DGA domain algorithm based on deep learning," *Journal of Shandong University(Natural Science)*, vol. 54, no. 7, pp. 106–112, 2019.

[27] Z. Ming, B. Xu, and B. Shuai, "A deep learning method to detect web attacks using a specially designed CNN," in *Proceedings of the International Conference on Neural Information Processing*, Bangkok, Thailand, November 2017.

[28] M. Antonakakis, R. Perdisci, and Y. Nadji, "From throw-away traffic to bots: detecting the rise of DGA-based malware," in *Proceedings of the 21th USENIX Security Symposium*, Bellevue, WA, USA, August 2012.

[29] Y. Lu, G. Liu, Z. Jiang tao, W. Liu, B. h. wen, and Y. Dai, "Improved algorithm for detecting the malicious domain name based on the convolutional neural network," *Journal of Xidian University*, vol. 1, no. 12, pp. 1–8, 2019.

[30] J. Saxe and K. Berlin, "eXpose: a character-level convolutional neural network with embeddings for detecting malicious url, file paths, and registry keys," 2017, https://arxiv.org/abs/1702.08568.

[31] S. Schiavoni, F. Maggi, and L. Cavallaro, "Phoenix: DGA-based botnet tracking and intelligence," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, Cham, Switzerland, March 2014.

[32] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modeling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 655–665, ACL, Baltimore, MD, USA, June 2014.

[33] L. Hung, Q. Pham, D. Sahoo, C. Steven, and H. Hoi, "URLNet: learning a URL representation with deep learning for malicious URL," 2018, https://arxiv.org/abs/1802.03162.

[34] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Processing Letters*, vol. 48, no. 3, pp. 1347–1357, 2018.

[35] Z. Wang, S. Li, B. Wang, X. Ren, and T. Yang, "A malicious URL detection model based on convolutional neural network," in *Proceedings of the International Symposium on Security and Privacy in Social Networks and Big Data*, pp. 34–40, Tianjin, China, September 2020.