Hindawi
Security and Communication Networks
Volume 2021, Article ID 4917016, 12 pages
https://doi.org/10.1155/2021/4917016

Research Article

A Novel Approach for Malicious URL Detection Based on the Joint Model

JianTing Yuan,[1] **YiPeng Liu**,[2] **and Long Yu**[3]

[1]*School of Software, Xinjiang University, Urumqi 830000, China*
[2]*School of Information Science and Engineering, Xinjiang University, Urumqi 830000, China*
[3]*Network Center, Xinjiang University, Urumqi 830000, China*

Correspondence should be addressed to Long Yu; yul_xju@163.com

Received 6 July 2021; Revised 2 October 2021; Accepted 22 November 2021; Published 13 December 2021

Academic Editor: Leandros Maglaras

The number of malicious websites is increasing yearly, and many companies and individuals worldwide have suffered losses. Therefore, the detection of malicious websites is a task that needs continuous development. In this study, a joint neural network algorithm model combining the attention mechanism, bidirectional independent recurrent neural network (Bi-IndRNN), and capsule network (CapsNet) is proposed. The word vector tool word2vec trains the character- and word-level uniform resource locator (URL) static embedding vector features. At the same time, the algorithm will also extract texture fingerprint features that can compare the content differences of different malicious web URL binary files. Then, the extracted features are fused and input into the joint neural network algorithm model. First, the multihead attention mechanism is used to extract contextual semantic features by adjusting weights and Bi-IndRNN. Second, CapsNet with dynamic routing is used to extract deep semantic information. Finally, the sigmoid classifier is used for classification. This study uses different methods from different angles to extract more comprehensive features. From the experimental results, the method proposed in this study improves the classification accuracy of malicious web page detection compared with other researchers.

1. Introduction

With the continuous improvement of the network environment, Internet applications have penetrated deeply into all aspects of life. Simultaneously, the vast Internet applications group also attracted many network attacks to make a profit through malware, spam, and phishing websites. According to the Check Point's report in 2020 [1], more than 100,000 malicious websites are used to steal users' personal information or cause damage to users' systems every day around the world. Kaspersky's report [2] stated that the number of malicious URLs identified by web antivirus components in 2020 was 173 million. Besides, the report also mentioned that malicious URLs accounted for 66.07% of the 20 most active malicious programs. With the emergence of more and more malicious websites, more and more individuals and companies will suffer immeasurable losses worldwide.

The web page represented by the malicious URL contains malicious interactive code, such as HTML tags [3], JavaScript (JS) [4], and Cascading Style Sheets (CSS) [5]. The attacker writes the source code containing malicious JS tags into the website, and thereby, the malicious code is executed while the user is visiting the website. For example, a remote download program is executed in the background when a user clicks on an advertisement implanted with malicious code by a hacker. The user terminal is finally controlled to collect user personal information. In addition, phishing websites are also the main battlefield for malicious URLs. The attacker establishes an illegal site and leads users into malicious web pages through inducements and other means to complete malicious acts such as network fraud. To dispel the user's precautionary psychology, the attacker will construct these websites very similar to the legitimate website, indistinguishable by the human eye. Accelerating the development of malicious URL detection has become an

essential task of network security in such a network environment.

So far, predecessors have proposed a lot of malicious URL detection methods. In previous research on malicious website detection, researchers usually manually extract one or more of the following features: web content feature HTML, JavaScript code, host information feature WHOIS, lightweight feature web URL, and visualization features, and then input them into the machine learning or heuristic learning system to detect malicious websites. For example, Kumar et al. [6] used an HTML parser and JavaScript simulator to extract web content features and input them into a heuristic system. Chu et al. [7] used domain-related information as the main feature and used machine learning for detection research. However, the feature engineering of machine learning technology is more cumbersome and relies on the subjective judgment of researchers. The emergence of deep learning has solved this well. Ren et al. [8] extracted the word embedding of the URL character to identify malicious URLs effectively. Peng et al. [9] added texture fingerprint features based on extracting URL and host information and then used a deep learning model for detection research. This study only focuses on URL features and uses deep learning techniques to detect and research malicious websites.

Designers generally design URLs as meaningful words to facilitate memory, and some meaningless words usually convey information in their character sequence. Therefore, we use word embedding and character embedding technology to extract the semantic features of URLs. Since the URLs generated by the same tool or organization have similar structures, we also extracted the URL texture fingerprint features (Section 3). Therefore, a joint neural network algorithm model was proposed to capture URL features. First, the attention mechanism is used to give higher weight to key features. Second, we used an improved independently recurrent neural network (IndRNN) [10] called the bidirectional IndRNN (Bi-IndRNN) model to encode the fusion feature information. Finally, the CapsNet is to extract high-level semantic features. Through experiments, it is found that the stacked CapsNet has made significant progress, and the joint model is a precious exploration. The innovations of this study are summarized as follows:

(1) We constructed a joint neural network algorithm model that combines the attention mechanism, Bi-IndRNN, and CapsNet for malicious URL detection

(2) To obtain more specific and natural features, we have integrated different malicious URL feature information to extract combined semantic and image information

(3) A series of comparative experiments show that the joint model proposed in this study achieves better performance than some state-of-the-art methods

The subsequent chapters of this study are organized as follows. Section 2 introduces the contributions of previous researchers to malicious URLs, Section 3 introduces the details of the proposed method, Section 4 explains the experimental results and analysis, and Section 5 summarizes this study.

2. Related Work

The main aim of malicious URL detection is to distinguish malicious URLs from benign URLs. Previous researchers proposed the methods for the problem of malicious URL detection which are mainly divided into the following categories: blacklist-, rules-, machine learning-, and deep learning-based detection.

2.1. Blacklist. The method based on the blacklist is to detect malicious websites, mark them, and store them in the database, which contains relevant information of the malicious URL. A global distributed URL blacklist service system based on P2P technology is proposed [11]. Contributors share the blacklist information on storage nodes, and the client uses a plug-in form to ensure the user's normal browsing experience. Fukushima et al. [12] proposed a blacklist system based on the reputations of the IP address block and registrars used by attackers. To discover more malicious websites actively, some researchers have proposed methods to expand the blacklist by analyzing the features of malicious websites. Akiyama et al. [13] used existing malicious URL search structure neighborhoods to find unknown malicious websites and verify them to expand the URL blacklist. Prakash et al. [14] proposed a prediction system composed of multiple heuristic components to generate new URLs. Then, regular expressions and hash maps are used to approximately match the URL to verify whether it is malicious. Compared to passively submitting URLs to the blacklist, this method can discover and verify malicious URLs from the same malicious source, but the limitations are also apparent. It is impossible to find newly emerging malicious domains, that is, there is no better generalization ability. Although the blacklist-based approach is easy to operate, data storage and update will face challenges when malicious websites are added with a considerable amount every day.

2.2. Rule Matching. Researchers have proposed a rule matching method to solve the above problem, which uses some features to formulate rules to filter malicious URLs. Cao et al. [15] proposed a rule matching method called Automated Individual Whitelist (AIWL), which automatically uses the Naive Bayes classifier to operate and maintain a list of the login user interface (LUI) that users are familiar with. This detection method will warn users when they visit untrusted websites or submit confidential information to these websites. Nguyen et al. [16] proposed a system that calculated six heuristic values similar to the Levenshtein distance between the domain name and the Google search engine spelling suggestion and weighted and added these values to determine whether it is a phishing website based on the threshold. Liu and Zhang [17] proposed a two-wheeled phishing page check method. The first round checks the domain name, URL, and e-mail of the current page, and if it exceeds the threshold, it is directly identified as a phishing page. If it does not exceed the second round, the password, link, and picture are checked. If all the checks do not exceed

the threshold, it is a regular page. However, this method is only used in the financial field. Shekokar et al. [18] proposed a two-stage phishing page detection scheme. The first stage uses the LinkGuard algorithm to analyze the difference between visual links (links rendered by the browser) and actual links (hidden in HTML). The second stage compares the similarity between suspicious web page snapshots and legitimate web pages by calculating the discrete cosine transform. Although this method does not need to maintain a vast database of malicious websites, it cannot detect unknown malicious URLs because the establishment of rules relies on existing malicious URLs. Moreover, it requires much subjective experience to analyze malicious web pages. The rule-based approach can find some more obvious malicious websites. Nowadays, the features of malicious web pages are diversified, and many rule-based methods are helpless.

2.3. Machine Learning. As big data become more and more popular, machine learning with generalization and resistance to actual attacks has become the mainstream detection method for malicious URLs. To implement a self-learning model, researchers must have enough malicious website data. Furthermore, the known sites are used to train the algorithm model, and the unknown sites are classified through the already trained algorithms model. After these steps, the model will have specific dynamic detection capabilities. Shahrivari et al. [19] proposed a method that used feature engineering to construct a dataset that extracts 30 features from URL, web page content, and host information; then, 12 machine learning methods such as random forest and decision tree are used to detect phishing websites. Crisan et al. [20] used word embedding to represent URL information and increase the performance of naive Bayes, logistic regression, and SVM models by adding general domain-specific features. This method abandons the selection of features from complex page content and simplifies the data processing process. However, machine learning methods require much functional design. Once these functions are known to malicious website designers, it is easy to bypass these security settings. Singhal et al. [21] used machine learning to classify malicious websites and proposed concept drift detection to find the difference in data distribution between the feature vectors of the old training dataset and the newly collected dataset. The purpose is to prevent attackers bypass the detection rules by changing the URL after realizing that the feature is extracted from the URL. The methods proposed by Eshete et al. [22] use machine learning algorithms for training and customize the corresponding algorithms to further improve the generalization ability of the method. First, seven machine learning algorithms are trained by extracting 39 features in three categories: URL, page-source, and social reputation. Then, the web page category is determined by the confidence-weighted majority vote classification algorithm.

Although machine learning can improve detection accuracy and has a certain generalization ability, manually extracting features is still a time-consuming and labor-intensive task and can only extract shallow features.

2.4. Deep Learning. Different from machine learning, deep learning can automatically extract high-dimensional features based on preprocessed data. After time verification, deep learning has also become the mainstream malicious URL detection method. The emergence of deep learning broke the deadlock of traditional machine learning algorithms. Deep learning can automatically extract features compared to machine learning's feature extraction method, which frees up the time of manual feature engineering. Wei et al. [23] proposed a method for malicious URL detection using the CNN. This method first extracts character-level features from the URL. Then, the CNN is used to extract features and classification. Bahnsen et al. [24] proposed feature extraction and classification method of malicious URLs based on long and short-term memory networks. This method analyzes 14 URL vocabulary features, such as subdomain length and URL entropy, to build feature engineering and LSTM for classification. The experimental results show that malicious webpage detection based on URL vocabulary features is more feasible than complete content analysis. Jiang et al. [25] proposed an online detection scheme based on a deep neural network to detect malicious URLs. This method maps URL and DNS into vectors and then uses the CNN to extract malicious features and train a classification model automatically. Nevertheless, this model can also be fine-tuned to make the model predictions more accurate. Das et al. [26] compared the application of a simple RNN, simple LSTM, and CNN-LSTM architecture to malicious URL classification in their research. After comparing accuracy, precision, and recall rate, the performance of CNN-LSTM architecture is better than the other two. The enlightenment of this research is as follows. Different models have different ideas for feature extraction. It is advisable to optimize the process of feature extraction by fusing models.

In general, deep learning technology has significantly improved the performance of malicious URL detection. Our method can process data faster than previous research results, which is essential in applying malicious URL detection tasks. In addition, the fusion of texture fingerprint features enables the model to have the ability to process URLs with complex structures, and the fusion of features enables the model to have better recognition accuracy. Experiments results show that our method improves the performance of malicious URL detection and classification. Although machine learning can improve detection accuracy and has a certain generalization ability, manually extracting features is still time-consuming and labor-intensive and can only extract shallow features.

3. Our Approach

3.1. Feature Analysis. The malicious behavior of malicious websites is generally manifested in the URL and website content. However, the method proposed by [27] bypassed

the website content, directly used URL to extract features and classify them, and achieved good experimental results. This study is inspired by this and only focuses on the website URL. To grasp the global features of malicious URLs, we extracted the texture fingerprint features of website URLs. However, this type of feature is only a superficial feature and does not fully reflect the essential attributes of URLs. So, we make a static analysis of the website URL to extract the semantic features of the website URL. In general, this study extracts two types of features: texture fingerprint features and semantic features.

3.1.1. Semantic Feature. By analyzing the malicious URLs published by PhishTank [28], Openphish [29], we found that the creators of some phishing websites usually imitate the content of regular pages and bind a similar domain name, such as "http://www.amazzonn.online." Besides, some special characters may be used to confuse users, such as "@" and "-". What is more, it confuses users by lengthening the string of meaningless characters or increasing the depth of the domain name (that is, the number of "."), such as "mlwdkaflzkpqccqdaxjuqlltyexdfcfuzufo-dot-cryptic-now-290917.ey.r.appspot.com." So, we extracted URL word features. First, symbolize the input URL and decompose the string into their constituent words. The symbolic description is shown in Figure 1.

To enable the symbolized data to be processed by the computer, it is necessary to embed the words obtained in the above steps and convert them into digital vectors containing the grammatical and semantic information of the words. The specific method is to embed the symbolized data into a $V \times D$ matrix and update it through backpropagation, where V represents the size of the vocabulary, and D is the dimension of word embedding. When we use word2vec to get the vectors of most words, meaningless words and symbols will confuse our model, so we also extracted URL character features. The process is similar to the process of word embedding. At this stage, we extracted two granular levels of embedding from the website URL: word level and character level.

3.1.2. Texture Fingerprint Feature. We also extracted visual features from the URL. In the experiment of Wang et al. [30], it was concluded that the same malicious web page family has similarities in texture fingerprints. In previous studies, Su et al. [31] and Yang and Wen [32] have proved the validity of grayscale images for deep learning models. Inspired by these conclusions, the URLs were also converted into grayscale images. The two-dimensional texture fingerprint features in the range of 8-bit unsigned integers are converted into effective texture fingerprint features corresponding to the grayscale image's gray value range.

Specifically, as shown in Figure 2, read the original data in binary form, and use each 8-bit read as a basic unit (fill it with 0 if the last read is less than 8-bit). Then, convert each basic unit to an unsigned integer, so that each integer value is guaranteed to be in the range of [0, 255]. Each integer is mapped to a grayscale image and represents the grayscale value of each pixel. "0" means pure white, and "255" means pure black. Finally, the gray value is stored in a fixed-width matrix.

3.1.3. Feature Fusion. To further improve the accuracy of detection, the three features of character-, word-level embedding vector, and texture fingerprint features have been fused. Given a sequence W_i represents the i^{th} word, C_i represents the i^{th} character of W_i, and P_i represents the i^{th} pixel of the grayscale image. The following formula can be used to express the joint vector:

$$X = [\text{Emb}(W_i), \text{Emb}(C_i), P_i], \tag{1}$$

Where "[]" represents the vector cascade, and $\text{Emb}(i)$ denotes the embedding of this i. After the features are fused, they are sent to the model for training and prediction.

3.2. Framework of the Model. A deep learning framework for detecting malicious URLs based on Bi-IndRNN and CapsNet is proposed in this study. The main structure of the framework is shown in Figure 3. First, the displayable characters and words are embedded in the multidimensional feature space using character embedding and word embedding components. The texture fingerprint feature of the URL is simultaneously extracted. Subsequently, merge the selected features and input to the attention mechanism, which assign probability weights to the mixed features to obtain features with higher weights. Next, an improved IndRNN called Bi-IndRNN is used to extract features from long sequences. We input the features extracted by the attention mechanism into Bi-IndRNN. The features extracted by Bi-IndRNN are input into CapsNet to establish high-level feature information. Finally, the sigmoid classifier is used to calculate the probability.

3.2.1. Attention Mechanism. The contribution of each joint vector to the feature expression of malicious URLs is different. As the attention mechanism can give higher weight to key features to highlight the impact of key features on downstream models, we stacked the attention mechanism on the top layer of the joint vector. Bahdanau et al. [33] first used the attention model in machine translation. The main task of the attention mechanism is to extract the most critical information for the model from a large number of given inputs by simulating the attention behavior of people to improve the efficiency of model training as much as possible while minimizing feature loss. At a macro level, the attention model can be understood as a mapping from a query to a series of key-value pairs. In essence, the attention mechanism is to perform a weighted summation of value. Then, query and key are used to calculate the weight coefficient of the corresponding value.

In this study, the multihead attention is introduced to structure a subset of URL high-dimensional features. The multihead attention is also based on query, key, and value, represented by $Q, K, V \in R^{n \times d}$ (n represents the number of URL features, and d represents the dimension of URL

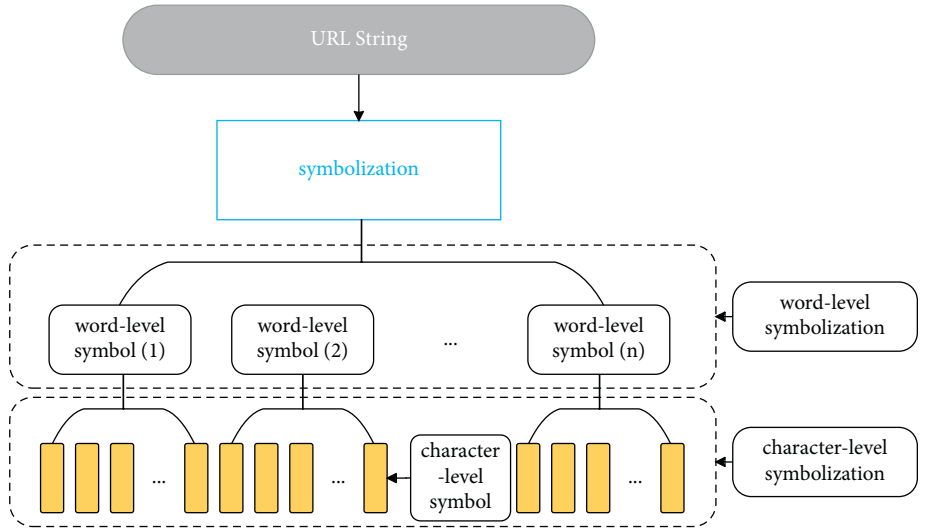

Figure 1: Symbolization process.

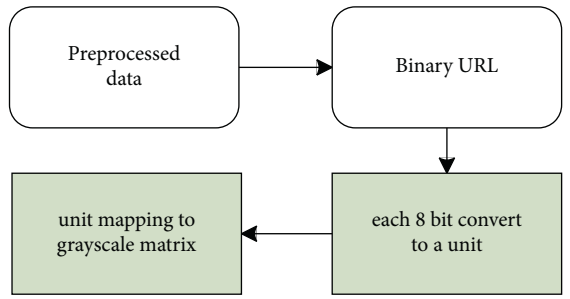

Figure 2: Malicious URL texture fingerprint feature analysis.

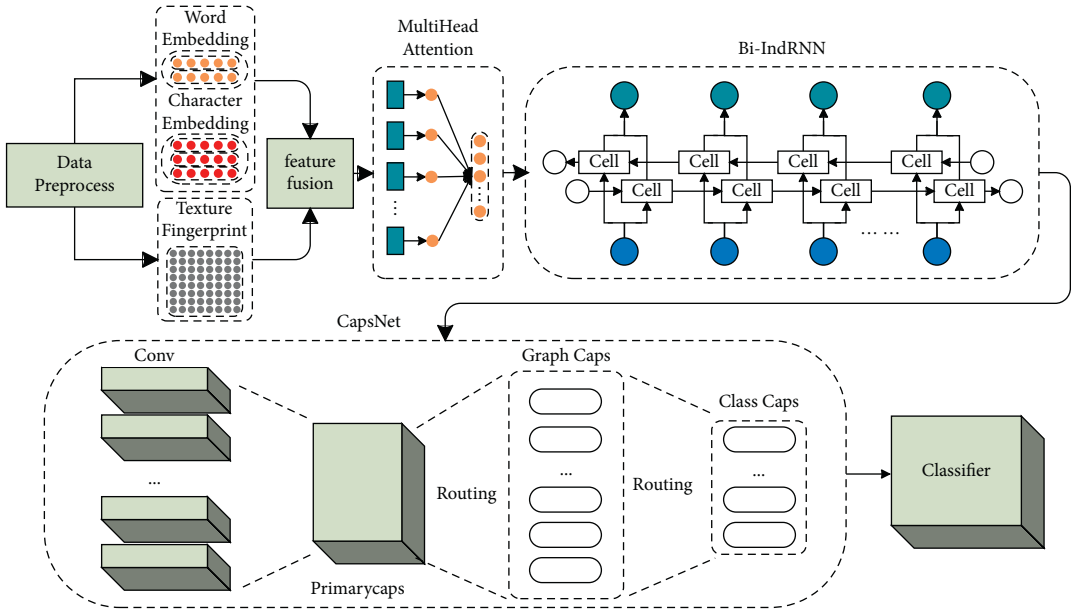

Figure 3: Architecture structure.

features), respectively, which will obtained by applying linear projections. Different from the general attention, multihead attention uses scaled dot-product attention to calculate the attention score. Given $X = \{x_1, x_2, \ldots, x_n\} \in R^{n \times d}$ represents the URL fusion feature vector, x_i represents the i^{th} feature vector, and input the x_i into the attention model:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(x_i * \left[x_1^T, x_2^T, \ldots, x_n^T\right]\right) * \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \text{softmax}\left(Q_i K^T\right)V. \tag{2}$$

The key of multihead attention is to use the above attention multiple times, and the number of "head" represents the number of times to perform the above attention. Then, it should be calculated as follows to obtain the attention of all URL feature vectors:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} * \left[x_1^T, x_2^T, \ldots, x_n^T\right]\right) * \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \text{softmax}\left(QK^T\right) = \text{softmax}\left(QK^T\right)V. \tag{3}$$

However, the linear projection of Q, K, and V calculated by each head is different. Take the multihead attention model of the i^{th} head with h heads as an example:

$$M_i(Q,K,V) = \text{Attention}\left(QW^Q, KW^K, VW^V\right), \tag{4}$$

Where $W^Q, W^K, W^V \in R^{n\times d/h}$. After h calculations, concatenate the calculation results:

$$M(Q,K,V) = \text{Concat}\left(M_1, M_2, \ldots, M_h\right). \tag{5}$$

Finally, calculate the weighted sum of the input joint vector X and the obtained attention M to obtain the input of the next layer, the feature vector v. After the above calculations, we can determine which information is more critical when Bi-IndRNN processes the current task. Give this important information a higher weight, so as to obtain as much information as possible for the current task from the URL joint vector.

3.2.2. Bi-IndRNN. Recurrent neural network (RNN) can effectively process data with sequence characteristics. However, the RNN training will face the problem of gradient disappearance and explosion due to long-distance dependence. As a variant of the RNN, long and short-term memory network (LSTM) can make it easier for the RNN to save information many steps ago, but it does not guarantee that gradients will not disappear or explode. In order to break through the situation at the time, Li et al. [10] proposed an independent recurrent neural network. This method effectively solves the problem of gradient disappearance and explosion because it can well apply ReLU and other nonlinear activation functions and can adjust the time-based gradient backpropagation. The IndRNN unit structure is shown in Figure 4. The hidden layer of IndRNN can be described as

$$h_t = \sigma\left(Wv_t + u \odot h_{t-1} + b\right), \tag{6}$$

where W, u, and b represent the input weight, recurrent weight, and bias, respectively, $\odot$ denotes the Hadamard product, and v_t denotes the input vector.

However, IndRNN can only obtain features through forwarding information when processing sequences to enable the model to integrate feature information better and have better modeling capabilities. The improved IndRNN called Bi-IndRNN has been used in this study. Bi-IndRNN is based on IndRNN and adds the idea of the bidirectional recurrent neural network (BRNN). That is, for each time t, the input will be given to two independent IndRNN units in the front and rear directions at the same time, and the output will be jointly determined by the two unidirectional IndRNN units.

The joint vector is a description of semantic and visual information, including important text structure and the spatial position distribution between characters. In order to enable the content represented by the joint vector to have more robust information representation capabilities, we use the Bi-IndRNN model to extract features from the joint vector. Given an feature vector $V = \{v_1, v_2, \ldots, v_t\}$ extracted from fusion feature by multihead attention, in the Bi-IndRNN we implemented, forward IndRNN $\overrightarrow{h}_t$ reads the feature sequence from v_1 to v_t, and backward IndRNN h_t reads the feature sequence from v_t to v_1. The hidden state expression can be expressed as follows:

$$\begin{aligned} \overrightarrow{h}_t &= \sigma\left(Wv_t + u \odot \overrightarrow{h}_{t-1} + b\right), \\ \overleftarrow{h}_t &= \sigma\left(Wv_t + u \odot \overleftarrow{h}_{t-1} + b\right). \end{aligned} \tag{7}$$

Next, we combine these two vectors as the output of Bi-IndRNN. In this way, each hidden state has information of the entire sequence, which is concentrated around the t^{th} sequence of the input vector. Then, the features vector extracted by Bi-IndRNN will be input into the capsule network to further extract deep features.

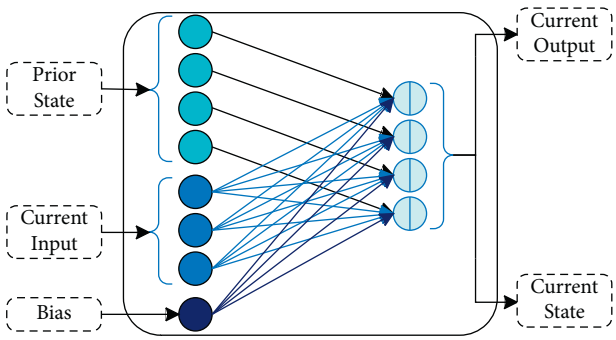

Figure 4: IndRNN unit structure.

3.2.3. Capsule Network. This study introduces the capsule network to establish advanced feature information. The extracted feature data can play a huge advantage when the capsule network is built on the top of the Bi-IndRNN layer. In order to solve some of the defects of the convolutional neural network to adapt to new deep learning tasks, Sabour et al. [34] proposed the capsule network in 2017. The capsule network is also a kind of neural network. The difference from the ordinary neural network is that the neurons of the capsule network are vectors instead of scalars. Each dimension of these vectors represents an attribute of the object. Therefore, the capsule network retains the posture information and spatial relationships between objects to the greatest extent. As part of the overall model, the structure of the capsule network is shown in Figure 3. First, input the features extracted from Bi-IndRNN to a standard convolution layer. The convolution operation is as follows:

$$\begin{aligned} z_l &= \sigma\left(W_1 \circ X_{l:\, l+k-1} + b\right), \\ Z^{(i)} &= \left[z_1, z_2, \ldots, z_{L-k+1}\right], \end{aligned} \tag{8}$$

where $\circ$ is the element-wise multiplication, b denotes the bias, and $W_1 \in R^{K \times d}$ denotes the convolutional filter, where the size of the convolutional filter is denoted by $K \times d$. That means the convolution operation is to slide the filter on a given input to extract features and collect them in a feature map.

Next is the capsule layer, which converts the feature map into a capsule through a group-convolution operation:

$$p_i = g\left(W_2 Z^i\right) \in R^V, \tag{9}$$

where V denotes the dimension of a capsule vector, Z^i represents the i^{th} dimension capsule vector, and function g means the nonlinear squash function, which expressed by the following formula:

$$g(x) = \text{squash}(x) = \frac{\|x\|^2}{1 + \|x\|^2} \frac{x}{\|x\|^2}. \tag{10}$$

Each capsule in the i^{th} layer in the network needs to predict the output $\hat{u}$ of the $(i+1)$ layer capsule separately:

$$\hat{u}_i = W_3 p_i. \tag{11}$$

Then, calculate the weighted sum of all prediction vectors to get the high-level capsule v:

$$v_j = g\left(\sum_i c_i \hat{u}_i\right), \tag{12}$$

where c_i is the coupling coefficient obtained by the dynamic routing algorithm. The capsule network can retain the most valuable information to the greatest extent and then save it completely and submit it to the upper capsule.

Finally, input the result obtained into the sigmoid classifier to get the final probability. So far, our model can complete the detection of malicious URL.

4. Results and Discussion

4.1. Experimental Set-Up. In Table 1, the attention layers, Bi-IndRNN layers, and CapsNet layers represent the number of attention, Bi-IndRNN, and CapsNet layers, respectively. The attention units and Bi-IndRNN units represent the number of multihead attention and Bi-IndRNN hidden layer units. The head denotes the number of heads of multihead attention. The capsule numbers and capsule dimensions denote the number and dimension of capsules, respectively. Our model uses the Adam optimizer with a default learning rate of 0.001.

4.2. Dataset. The dataset in this study consists of benign and malicious instances. We obtained a collection of benign URLs from the top rankings of Alexa verified by Google Safe Browsing, and the collection of malicious URLs was obtained from public websites, such as host-file.net and phishtank.com. In the end, 32,378 benign URLs and 33,549 malicious URLs were obtained.

4.3. Evaluation Indicators. We use five-fold cross-validation. The dataset is equally divided into five parts, four parts of which are used as training data and 1 part used as test data, and experiments are carried out in turn. Accuracy (ACC), precision (P), recall (R), and F score (F) are used to evaluate the classification results. Before the evaluation, it is necessary to count the number of experimental results correctly classified as malicious (TP) and benign (TN) samples and the number of incorrectly classified as malicious (FP) and benign (FN) samples. The evaluation is calculated as follows:

$$\begin{aligned} \text{ACC} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ P &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ R &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ F &= 2 \times \frac{P \times R}{P + R}. \end{aligned} \tag{13}$$

4.4. The Influence of Model Parameters on Experimental Results. In the model training process, we found that the influence of model parameters on experimental results is

Table 1: Experimental set-up.

Parameter	Value
Attention layers	1
Bi-IndRNN layers	2
CapsNet layers	1
Attention units	200
Heads	8
Bi-IndRNN units	64
Capsule dimensions	32
Capsule numbers	2
Optimization algorithm	Adam

quite apparent. Appropriate parameter settings will have a positive effect on model training and classification results. To determine these parameters and obtain the optimal classification results, we test these variable parameters, such as feature types, feature dimensions, under the same dataset, and determine the optimal parameters based on the evaluation indicators.

To determine which feature type to use to have the best classification performance, we first use three types of features: character embedding, word embedding, and texture fingerprint features to test separately and then combine these three features for testing. The results are given in Table 2. It can be concluded that using character and word embedding alone for classification can have good performance, reaching 99.82% and 99.89% recall rates, respectively. In contrast, the performance of the texture fingerprint classifier is slightly weaker, reaching a recall rate of 97.48%. It can also be concluded from the table that although the use of character embedding features can get good results with an accuracy of 99.74%, the method of combining the three features has a stable improvement in various evaluation indicators.

The dimension of the feature vector also has a specific impact on the experimental results. We used different dimensions as variable parameters to determine the feature dimensions and divided them into six groups of varying feature dimensions for comparison. As shown in Figure 5, in these six sets of experiments, ten feature dimensions are added each time the next set of experiments is performed. When the feature dimension increased from 90 to 130, all evaluation indicators increased, but the results obtained by continuing to increase the feature dimension are not ideal. When the dimensionality increases from 130 to 140, all other evaluation indicators decrease except for the slight recall rate increase, and the precision decrease is more pronounced. From this, we have determined that the dimension of the feature is 130.

4.5. The Necessity of Model Components. In this part, several sets of experiments are designed to verify the effectiveness of each part of the model. After comparing the three attention mechanisms, we found that these attentions have good performance. As shown in Figure 6, the accuracy of self-attention, hierarchical attention, and multihead attention can reach 99.75%, 99.67%, and 99.78%, respectively. However, the multihead attention

Table 2: The influence of each feature on the experimental results.

Feature types	Character (C)	Word (W)	Fingerprint (F)	C + W + F
ACC (%)	99.74	99.69	97.47	99.78
P (%)	99.64	99.48	97.35	99.65
R (%)	99.82	99.89	97.48	99.90
F (%)	99.73	99.68	97.42	99.78

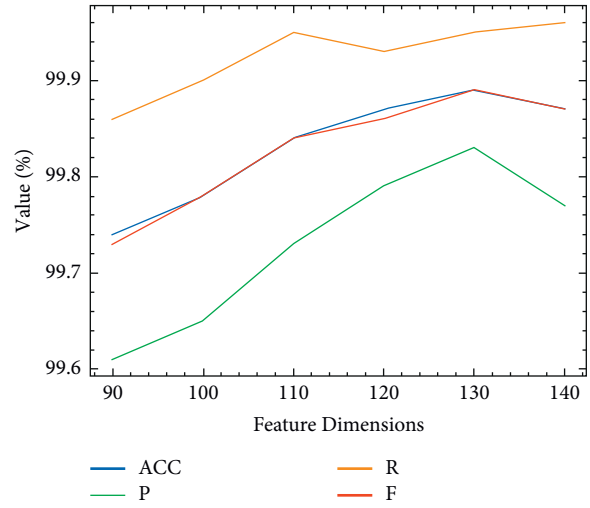

Figure 5: Experimental results of feature dimensions.

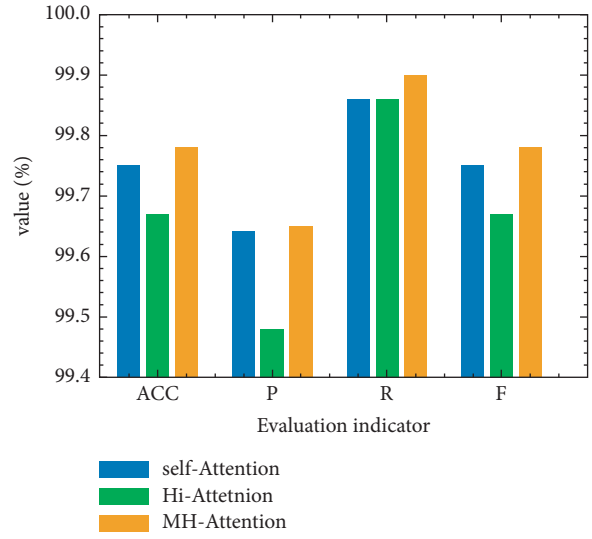

Figure 6: Experimental results of different attention mechanisms comparisons.

mechanism has improved significantly in various evaluation indicators, and the recall rate can reach 99.90%, which is helpful for the detection and classification of malicious URLs. So, the multihead attention had been used as a component of the model in this study.

Our model combined the attention mechanism, Bi-IndRNN, and CapsNet components. In order to verify the effectiveness of each component, three other models were designed. The three groups of models in the table are

(i) Attention-based IndRNN (AIR): extract feature information and classify through the attention-based IndRNN sequential model without CapsNet.

(ii) Attention-based CapsNet (ACaps): extract feature information and classify through the attention-based CapsNet sequential model without IndRNN.

(iii) IndRNN + CapsNet (IRCaps): use IndRNN and CapsNet sequential models for detection and classification without the attention mechanism.

Such a comparative experiment allows us to see the contribution of each component in the model. As given in Table 3, the AIR model without CapsNet is lower in accuracy, precision, and recall rate than the model used in this study, which shows the effectiveness of the capsule network. For ACaps without IndRNN, the result is similar to the AIR model, and all evaluation indicators are also lower than our model. In addition, the IRCaps model removes the attention mechanism, and the result is as we expected. The model performance is not as good as our model due to the inability to select more helpful information, which also illustrates the necessity of the attention mechanism.

In order to verify that our proposed model is more suitable for malicious URL detection and classification, a set of experiments had been designed to compare with methods by using other deep learning models. The experimental results are shown in Table 4. In this experiment, we fixed the hyperparameters and input the same data set into different models under the same experimental environment to verify the improvement of our model.

These methods have good performance for malicious URL detection and classification. Wanda and Jiang [35] also use character embedding technology and a single CNN architecture to extract features and classify, with a precision of 99.7%. Nevertheless, it is slightly inferior in accuracy and F value. Bahnsen et al. [24] and Liang et al. [36] used LSTM and Bi-LSTM models, respectively, and it can be concluded that the Bi-LSTM model with an accuracy of 99.74% performs slightly better than LSTM. In Wang's [37] method, after fusing the host features and URL information features, Bi-IndRNN is used for detection and classification, and finally, a recall rate of 99.93% is obtained. Besides, a separate attention, IndRNN model experiment, comes to similar results. Furthermore, CapsNet can save more features that can play its advantages. A single CapsNet [38] surpasses other single models in some evaluation indicators. However, the model proposed in this study combines these advantages and can outperform the previous model in various evaluation indicators. By comparing LSTM and Bi-LSTM, IndRNN, and Bi-IndRNN, it can be concluded that using a bidirectional network has better model performance than using a unidirectional network. Compared with other models, our model has improved in all indicators, and the accuracy and recall rates have reached 99.78% and 99.98%.

4.6. The Cost of the Model. To study the time cost of the proposed model, we conducted comparative experiments on the methods proposed by previous researchers. The experiment is divided into five groups. Each group of experiments uses early stop to prevent the model from overfitting, so the epoch of each model training is not the same. The experiment tested the average time required for a single epoch of each model, the average total time required to train a complete model, the trainable parameters of a model, and the test accuracy. The parameters of the hardware used in the experiment are given in Table 5, and the time cost experimental results are given in Table 6.

It can be known from the experimental results that the classic model can also achieve good results in a short time. For example, an attention-based Bi-LSTM model called AB-LSTM proposed in [8] can get 99.69% of the test accuracy. In pursuit of higher accuracy, researchers have proposed more complex models to detect malicious URLs. The TException method proposed in [39] uses multiple TException Blocks composed of 1d convolutional, batch normalization, Max-pooling, ReLU layer, and deep neural network (DNN) layers to perform feature processing on character-level and word-level URLs. This method uses multiple batch normalization layers to speed up the training, but this will also reduce the expression ability of the subsequent activation function, resulting in a limited improvement in accuracy. Both the attention-based CNN-LSTM (ATT-CNN-LSTM) method proposed in [40] and the CNN and attention-based hierarchical RNN (ATT-CNN-HRNN) method proposed in [41] combine CNN and RNN related methods, which can effectively extract relevant features and achieve malicious URL detection. It can be seen from Table 2 that compared with other new methods, our method does require a longer time in the training of a single epoch, which is caused by the routing protocol algorithm in the internal loop of the capsule network. However, the smaller training parameters make our method converge faster, have the same order of magnitude total training time as other advanced algorithms, and have higher test accuracy.

Table 3: Necessity of model components.

Total	ACC (%)	P (%)	R (%)	F (%)
AIR	99.62	99.64	99.58	99.89
ACaps	99.62	99.61	99.54	99.83
IRCaps	99.76	99.70	99.81	99.95
Our	99.87	99.79	99.93	99.89

Table 4: The influence of each algorithm on the experimental results.

Total	ACC (%)	P (%)	R (%)	F (%)
Attention	99.58	99.7	99.44	99.57
CapsNet	99.74	99.52	99.96	99.74
CNN	99.62	99.7	99.1	99.62
LSTM	99.65	99.8	99.49	99.64
Bi-LSTM	99.74	99.58	99.89	99.73
IndRNN	99.52	99.59	99.44	99.52
Bi-IndRNN	99.68	99.43	99.93	99.68
Our	99.89	99.83	99.95	99.89

Table 5: Hardware parameters.

Hardware name	Model
CPU	Xeon E5-2678 v3
Video memory	11G
RAM	62G
GPU	NVIDIA GeForce RTX 2080 Ti
Hard drive	100G SSD

Table 6: Experimental results of model cost.

Total	Average time to train an epoch(s)	Average time to train a model(s)	Trainable parameters	Test accuracy (%)
AB-LSTM	19.58	860.67	339,477	99.75
TEXCEPTION	30.45	1710.32	2,396,869	99.66
ATT-CNN-LSTM	65.98	4095.43	28,434,807	99.76
ATT-CNN-HRNN	90.23	5580.65	25,860,875	99.79
Our	120.26	5280.75	7,124,529	99.89

5. Conclusions

This study proposed a joint neural network algorithm model combining the attention-based bidirectional independent recurrent network (Bi-IndRNN) and capsule network (CapsNet) to identify and detect malicious URLs. It can be concluded from the experiment that the performance of this method to detect malicious URLs is significantly better than these of a single deep neural network and a shallow neural network. The key to this study is to use the generated word vector model word2vec to train to obtain URL words and character vector features, extract the texture fingerprint features of the URL, and fuse the three features. Then, extract key features are based on the weight of the multihead attention mechanism and Bi-IndRNN, and finally, use the capsule network to build high-dimensional features and classify them. Besides, in the same experimental environment, we compared different feature types and dimensions, different model components, and algorithm models. In summary, the method proposed in this article can effectively improve the detection efficiency and accuracy of malicious URLs.

It can be improved, although the method in this study had performed well. In the follow-up process, we will consider integrating dynamic and static features to verify its effectiveness. At the same time, we will continue to update the model, integrate new components into the system, and optimize the time cost of the model to achieve a more excellent method.

Data Availability

The data used to support the findings of this study have been deposited in the GitHub repository (https://github.com/yipeng-liu-rep/malicious-url-data).

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the Xinjiang Autonomous Region Key R&D Project (2021B01002), National Natural Science Foundation of China (U2003208), and CERNET Innovation Project (NGII20190412).

References

[1] Check Point, *Cyber Security Report 2021*, Check Point, Tel Aviv-Yafo, Israel, 2021.

[2] Kaspersky, "Kaspersky security bulletin 2020. statistics," 2020.

[3] J. Saxe, R. Harang, C. Wild, and H. Sanders, "A deep learning approach to fast, format-agnostic detection of malicious web content," in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, pp. 8–14, IEEE, San Francisco, CA, USA, May 2018.

[4] P. N. Hiremath, "A novel approach for analyzing and classifying malicious web pages," Doctoral dissertation, University of Dayton, Dayton, OH, USA, 2021.

[5] B. Chen and Y. Shi, "Malicious hidden redirect attack web page detection based on CSS features," in *Proceedings of the 2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, December 2018.

[6] K. P. Kumar, N. Jaisankar, and N. Mythili, "An efficient technique for detection of suspicious malicious web site," *Journal of Advances in Information Technology*, vol. 2, no. 4, pp. 217–221, 2011.

[7] W. Chu, B. B. Zhu, F. Xue, X. Guan, and Z. Cai, "Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs," in *Proceedings of the 2013 IEEE international conference on communications (ICC)*, pp. 1990–1994, IEEE, Budapest, Hungary, June 2013.

[8] F. Ren, Z. Jiang, and J. Liu, "A Bi-directional LSTM model with attention for malicious URL detection," in *Proceedings of the 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, IEEE, Chengdu, China, December 2019.

[9] Y. Peng, S. Tian, L. Yu, Y. Lv, and R. Wang, "A joint approach to detect malicious URL based on attention mechanism," *International Journal of Computational Intelligence and Applications*, vol. 18, no. 3, Article ID 1950021, 2019.

[10] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): building a longer and deeper rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5457–5466, IEEE, Salt Lake City, UT, USA, June 2018.

[11] J. G. Göbel, B. Stock, P. Trinius, and F. Freiling, *Blacklisting Malicious Websites Using Peer-To-Peer Technology*, University of Mannheim, Mannheim, Germany, 2010.

[12] Y. Fukushima, Y. Hori, and K. Sakurai, "Proactive blacklisting for malicious web sites by reputation evaluation based on domain and IP address registration," in *Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 352–361, IEEE, Changsha, China, November 2011.

[13] M. Akiyama, T. Yagi, and M. Itoh, "Searching structural neighborhood of malicious urls to improve blacklisting," in *Proceedings of the 2011 IEEE/IPSJ International Symposium on Applications and the Internet*, pp. 1–10, IEEE, Munich, Germany, 2011, July.

[14] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *Proceedings of the 2010 IEEE INFOCOM*, pp. 1–5, IEEE, San Diego, CA, USA, 2010, March).

[15] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in *Proceedings of the 4th ACM workshop on Digital identity management*, pp. 51–60, ACM Library, Alexandria, VA, USA, October 2008.

[16] L. A. T. Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen, "A novel approach for phishing detection using URL-based heuristic," in *Proceedings of the 2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, pp. 298–303, IEEE, Da Nang, Vietnam, 2014, April.

[17] Y. Liu and M. Zhang, "Financial websites oriented heuristic anti-phishing research,"vol. 2, pp. 614–618, in *Proceedings of the 2012 IEEE 2nd international conference on cloud computing and intelligence systems*, vol. 2, IEEE, Hangzhou, China, November 2012.

[18] N. M. Shekokar, C. Shah, M. Mahajan, and S. Rachh, "An ideal approach for detection and prevention of phishing attacks," *Procedia Computer Science*, vol. 49, pp. 82–91, 2015.

[19] V. Shahrivari, M. M. Darabi, and M. Izadi, *Phishing Detection Using Machine Learning Techniques*, Sharif University of Technology, Tehran, Iran, 2020.

[20] A. Crisan, G. Florea, L. Halasz, C. Lemnaru, and C. Oprisa, "Detecting malicious URLs based on machine learning algorithms and word embeddings," in *Proceedings of the 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 187–193, IEEE, Cluj-Napoca, Romania, September 2020.

[21] S. Singhal, U. Chawla, and R. Shorey, "Machine learning & concept drift based approach for malicious website detection," in *Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 582–585, IEEE, Bengaluru, India, January 2020.

[22] B. Eshete, A. Villafiorita, and K. Weldemariam, "Binspect: holistic analysis and detection of malicious web pages," in *Proceedings of the International conference on security and privacy in communication systems*, pp. 149–166, Springer, Berlin, Heidelberg, 2012, September.

[23] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, "Accurate and fast URL phishing detector: a convolutional neural network approach," *Computer Networks*, vol. 178, Article ID 107275, 2020.

[24] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonzalez, "Classifying phishing URLs using recurrent neural networks," in *Proceedings of the 2017 APWG Symposium on Electronic Crime Research (eCrime)*, pp. 1–8, IEEE, Scottsdale, AZ, USA, April 2017.

[25] J. Jiang, J. Chen, K.-K. R. Choo et al., "A deep learning based online malicious URL and DNS detection scheme," *International Conference on Security and Privacy in Communication Systems*, vol. 238, pp. 438–448, 2017.

[26] A. Das, A. Das, A. Datta, S. Si, and S. Barman, "Deep approaches on malicious URL classification," in *Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, Kharagpur, India, 2020, July.

[27] M.-Y. Kan, "Web page classification without the web page," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pp. 262-263, ACM Library, Portland, Oregon, USA, May 2004.

[28] Phishtank, http://www.phishtank.com/.

[29] Openphish, https://openphish.com/.

[30] H.-H. Wang, L. Yu, S.-W. Tian, Y.-F. Peng, and X.-J. Pei, "Bidirectional LSTM malicious webpages detection algorithm based on convolutional neural network and independent recurrent neural network," *Applied Intelligence*, vol. 49, no. 8, pp. 3016–3026, 2019.

[31] J. Su, V. D. Vasconcellos, S. Prasad, S. Daniele, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 664–669, Tokyo, Japan, July 2018.

[32] M. Yang and Q. Wen, "Detecting android malware by applying classification techniques on images patterns," in *Proceedings of the 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 344–347, IEEE, Chengdu, China, April 2017.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the ICLR 2015: International Conference on Learning Representations 2015*, ACM, Portland, OR, USA, September 2014.

[34] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3856–3866, 2017.

[35] P. Wanda and H. J. Jie, "URLDeep: continuous prediction of malicious URL with dynamic deep learning in social networks," *International Journal on Network Security*, vol. 21, no. 6, pp. 971–978, 2019.

[36] Y. Liang, J. Deng, and B. Cui, "Bidirectional LSTM: an innovative approach for phishing URL identification," in *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 326–337, Springer, Sydney, Australia, June 2019.

[37] H. Wang, S. Tian, L. Yu, X. Pei, and Y. Peng, "An algorithm based on Bi-IndRNN for analysis and detection of malicious URL," *Journal of Xinjiang University (Natural Science Edition)*, vol. 49, no. 1, pp. 174–181, 2019.

[38] Y. Huang, J. Qin, and W. Wen, "Phishing URL detection via capsule-based neural network," in *Proceedings of the 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, IEEE, Xiamen, China, October 2019.

[39] F. Tajaddodianfar, J. W. Stokes, and A. Gururajan, "Texception: a character/word-level deep learning model for phishing URL detection," in *Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2857–2861, IEEE, Barcelona, Spain, 2020, May.

[40] Y. Peng, S. Tian, L. Yu, Y. Lv, and R. Wang, "Malicious URL recognition and detection using attention-based CNN-LSTM," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 11, pp. 5580–5593, 2019.

[41] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing URL detection via CNN and attention-based hierarchical RNN," in *Proceedings of the 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 112–119, IEEE, Rotorua, New Zealand, 2019, August.