Received November 28, 2020, accepted December 30, 2020, date of publication January 6, 2021, date of current version January 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049625

Malicious URL Detection Based on a Parallel Neural Joint Model

JIANTING YUAN[1], GUANXIN CHEN[2], SHENGWEI TIAN[1], AND XINJUN PEI[1]
[1]School of Software, Xinjiang University, Ürümqi 830000, China
[2]School of Information Science and Engineering, Xinjiang University, Ürümqi 830000, China

Corresponding author: Shengwei Tian (tianshengwei@163.com)

This work was supported by the China Education and Research Network (CERNET) Innovation Project under Grant NGII20190412.

ABSTRACT A parallel neural joint model algorithm is proposed for the analysis and detection of malicious Uniform Resource Locator (URL). By detecting and analyzing malicious URL's characteristics, the semantic and visual information will be extracted. First, a visualization algorithm is used to realize the visualization of the URL mapping to a gray image with texture characteristics. Second, the lexical feature and character feature of URL are extracted and further processed through word vector technology. These extracted features are transformed into lexical embedding vectors and character embedding vectors. To combine the texture features with text features, a parallel joint neural network combining capsule network (CapsNet) and independent recurrent neural network (IndRNN) is utilized to capture multi-modal vectors of visual and semantic information synchronously. The last layer utilizes the attention mechanism to further filter the deep features extracted from the overall network while concentrating on effective features improving the classification accuracy and analyzing and detect malicious URLs. Based on the experimental results, it is demonstrated that this algorithm has higher accuracy compared to the traditional algorithms.

INDEX TERMS Malicious URL, cybercrime, capsule network, independent recurrent neural network, attention.

I. INTRODUCTION

Since the advent of the Internet, cyberattacks have emerged endlessly. According to a security report released by Microsoft in 2019 [1], phishing is still the primary method for attackers. Furthermore, the report indicated a continual increase in phishing attacks. In 2018, the number of phishing attacks increased by 250% per month. In the foreseeable future, phishing attacks will still be a problem. Since people always make correct judgments and decisions when facing the temptation of cybercriminals in every possible way, to identify these malicious websites effectively and quickly has been an essential issue in the security industry.

The attacker usually builds a website similar to the target or embeds the exploit code of browser vulnerabilities on the webpage. It tricks the victim into clicking on these links to obtain the victim's information or control the victim's computer. There have been numerous detection methods proposed for phishing websites so far. These methods are based on certain features of the website to distinguish the website. The features can be divided into the following categories:

The associate editor coordinating the review of this manuscript and approving it for publication was Alessandra De Benedictis.

URL-based

- URL:protocol://hostname[:port]/path/[;parameters][?query]#fragment, the above is the structure of a standard URL. Mainly, the host, path, and parameters are used, and there is a certain logical relationship between the words appearing in each part and the resource represented by the URL. Kan [2] research indicated that the URL carries rich information about the properties of the website using URL only for website classification.
- Domain: The domain name is composed of multiple words and period separators and can be considered as the only mnemonic for one IP address. A normal domain name can often represent the belonged organization. Yadav *et al.* [3] calculated the distance between the URL labels and the phishing word.

Host-based:

- Whois: Whois is a protocol used to search for information such as domain names, IP addresses, and domain name owner information on the Internet. For malicious websites, the domain name is often not long registered. Chu *et al.* [4] uses the domain name age as a feature for identifying malicious websites.

- DNS: DNS is a distributed database storing mapping records of domain names and IP addresses. For each record, it contains the domain name, type, value, classification, and survival period. In general, the survival period of malicious website domain names is shorter than that of normal organization domain names. Dahu *et al.* [5] extracts features from DNS requests and responses to find malicious websites. Pereira *et al.* [6] Pereira proposed a WordGraph method to extract the word dictionary used by the DGA algorithm from DNS traffic.

Content-based:

- HTML: HTML is a markup language consisting of a series of tags and all the resources of a page. Li *et al.* [7] uses html features such as hidden information, form, and the number of links to achieve better detection results.
- Code: javascript originally exists for better interface interaction effects, however, it can be used by attackers to do more things. Hou *et al.* [8] counted 154 JavaScript functions for phishing web-page detection. Fang *et al.* [9] uses word vectors to map the bytecode of the js code into a multi-dimensional digital vector.
- Visual: It measures the similarity of two websites by detecting the visual elements of web pages. Wenyin *et al.* [10] first proposed this technique to calculate the similarity of two pages by measuring web page layout, frame, block-level, and overall style. Dalgic *et al.* [11] proposed to utilize SPM to count the features of a website screenshot.

Usually, the victims are attacked since they clicked on the URL of the phishing website posted by the attacker. Therefore, it is necessary to prompt the user that the URL points to a phishing website before the user clicks. The URL-based method is significantly faster than other methods because it does not require page parsing. Most companies use Blacklist to identify malicious URLs. This technique is easily be breached by an attacker. Through continuous testing, the attacker can find out which keywords have been blacklisted. Several URL classification studies based on machine learning algorithms need to pass a complex feature selection process. Most of the ultimately selected features are text information and host information. The text features are mainly based on the researcher 's subjective, which is not particularly convincing as the feature representation of the URL. Compared to the automatic feature extraction process of deep learning, the feature extraction process of machine learning is a bit clumsy. In Bahnsen *et al.* [12] research, first, the neural network model was utilized to classify URLs and excellent performance was represented. In this study, we focused on using deep learning technology and merely the URL to classify malicious websites.

To better learn the URL's hidden information, character embedding technology is introduced to extract the potential semantic information, and the URL is converted into a gray image. The order of characters in the URL is informative. Normally, numerous long URLs of the same class have the same or similar character sequence. A new technique of malware visualization based on image processing technology was proposed by Nataraj *et al.* [13] indicating that the image texture features are available. Through observation, it can be found that the URLs used by the same organization or generated by the same phishing attack tool have a similar structure. Thus, a parallel joint neural network model is proposed able to capture URLs' visual and semantic information. We use CapsNet [14] to encode image texture features and indRNN [15] for encoding the URL text features. Then, we merge the two extracted features and use the attention mechanism to further filter them while focusing on the effective features enhancing the classification accuracy.

In the present work, we make the following contributions:

- We constructed a parallel joint neural network model simultaneously capturing the semantic and visual information of URLs.
- Our proposed model can automatically learn the URLs' feature representation from data and avoid the manual work of feature engineering.
- A series of controlled experiments verified our model's usability and good performance.

The remaining of the paper is organized as follows. Section II introduces the previous research on malicious URLs. In Section III, the methodology is presented including feature extraction and classification. Section IV explains the results and analysis of experiments. Section V summarizes the paper.

II. RELATED WORK

At present, the popular detection methods of malicious URLs mainly include Blacklist, machine learning, rule matching, and deep learning-based detection methods.

Kührer *et al.* [16] analyzed the IP addresses and domain names of multiple public blacklist data sets. They found that parked domains in the Blacklist can constitute a considerable number of blacklist entries, hence, they developed a graph-based approach to recognize the sinkholes within the blacklists. To increase the detection range of blacklist technology, the existing blacklists were expanded by some researchers. In the PhishNet system, built by Prakash *et al.* [17], five heuristics algorithms were used to list simple combinations of known phishing sites for finding new phishing URLs. This system first dissected known malicious URLs into multiple components and then detected new unknown malicious URLs data based on the similarity. An automatic blacklist generator (AutoBLG) was proposed by Sun *et al.* [18] to automatically generate new malicious URLs based on the original blacklist to expand the original blacklist set.

The performances of blacklist detection depend on the size of the original list, moreover, it takes a long time to expand the data set, therefore, some researchers have proposed a rule matching-based method. Rule matching is a text structure analysis of known malicious URLs able to identify a new unknown malicious URL based on the matching rule. SpyProxy [19] is a proxy-based anti-malicious website tool. A set of matching rules was designed by Moshchuk in the SpyProxy to match the intercepted content transmitted from the web server to the browser. A system was proposed [20] known as Cujo for automatic detection and prevention of download attacks. It is embedded in a web proxy to inspect web pages and identify malicious code in pages. Cujo automatically establishes rules based on detection results and matches malicious code based on the rules to prevent malicious transmission. A content-based phishing website detection tool Cantina was designed and implemented by Zhang *et al.* [21], for the IE browser. The tool analyzes the search results, TF-IDF values, and other statistical characteristics and develops relevant rules to determine malicious webpages.

The above methods cannot detect unknown malicious URLs well since the setting of blacklist and rules is relied on known malicious URLs and their update is difficult and not timely enough. To solve this problem, methods have been proposed based on machine learning and deep learning. Vanhoenshoven *et al.* [22] utilized multiple machine learning algorithms such as K-nearest neighbor, Decision Tree, Naive Bayes, Random Forests, and Support Vector Machine to detect malicious URLs in datasets with three different statistical characteristics. Random Forest and Multi-Layer Perceptron perform best in this regard. Gawale *et al.* [23] designed and implemented a malicious URL detection system based on short URLs and malicious content. The system detected malicious URLs in Twitter texts through multiple features such as original URLs, similar texts, related URLs, and the following rates. Daeef *et al.* [24] used various machine learning algorithms to classify URLs based on the n-gram features of URLs. Jayakanthan *et al.* [25] proposed a malicious URL detection method based on the combination of the EPCMU and the Naive Bayes algorithm. Multiple URL features were used in this method, including the number of unique characters, the number of characters '/' or '@', and whether they are blacklisted, to create a feature vector. Azeez *et al.* [26] utilized the naive Bayes algorithm to detect malicious URLs based on the grammatical, lexical, host, and other of the URL embedded in the email.

However, these traditional machine learning algorithms need to manually extract the features of URLs. The feature vectors extracted in this way can only express the shallow features of the data, but not the buried features. Moreover, the methods of manual statistical features will consume a huge deal of time in feature extraction and selection. Bahnsen [12] compared two techniques, the Random Forests (RF) algorithm utilizing 14 lexical and statistical features of URL as an input and the Recurrent Neural Network algorithm based on character embedding. The ultimate results also indicated that the RNN algorithm is more appropriate for URL classification than RF, however, the disadvantage is that RNN requires more data for training. A deep learning model was proposed by Saxe *et al.* [27] based on multi-core convolution, eXpose utilizing a combination of character embedding and convolutional neural networks (CNN) to extract the malicious URLs features and detect the malicious URLs. The experimental results indicated that eXpose outperforms those manual feature-based models. A malicious domain name detection method was presented by Woodbridge *et al.* [28] based on long short term memory network (LSTM) utilizing the LSTM model to automatically extract the context features of malicious domain names, and completed the DGAs classification. However, this LSTM model is not very effective in discovering some families with very extraordinary structures.

In total, malicious URL detection techniques utilizing deep learning models generally yield better results. Compared to the above method, the feature processing part of our technique consumes less time, which is more advantageous in the identification of unknown malicious URLs. Moreover, the URLs' visual semantic information is combined to enhance predictive accuracy and obtain better performance in identifying malicious URLs with complex structures.

III. OUR APPROACH

A. FEATURE

The accuracy of classification is directly enhanced by an appropriate feature extraction method. As shown in Fig 1, our feature processing has two parts in total. By character embedding and word embedding components, displayable characters and words are embedded into a multidimensional feature space encoding the original URL into a two-dimensional tensor. The image component converts the URL into a grayscale image.

1) SEMANTIC FEATURE

Attackers often utilize the "domain impersonation" [1] method to trick victims by replacing a letter in a domain name with similar letters. For example, replace 'google' with 'goog1e'. There are several more such deception techniques. It is essential to find a way to express the characters and learn hidden information better on the computer to insert the URL into the model for training. One-hot is a basic vector method representing the characters in the URL through a vector of the number of characters. Only the item corresponding to the character in the vector is 1, and all other items are zero. Nevertheless, for one-hot, the high dimensionality of data is a problem, and the matrix composed of vectors corresponding to characters is too sparse. Displayable characters are embedded into a $s \times m$ floating-point matrix to better get the hidden information of the letters in URL, where s denotes the number of displayable characters, and m refers to the dimension of the embedded vector. This can encode characters into dense real-valued vectors. Meaningful words are used in URL both by a developer and an attacker. Hence, the words in the URL are also mapped to a floating-point matrix utilizing word embedding methods.

These two matrices of the embedded layer are also optimized with the rest of the model through backpropagation.

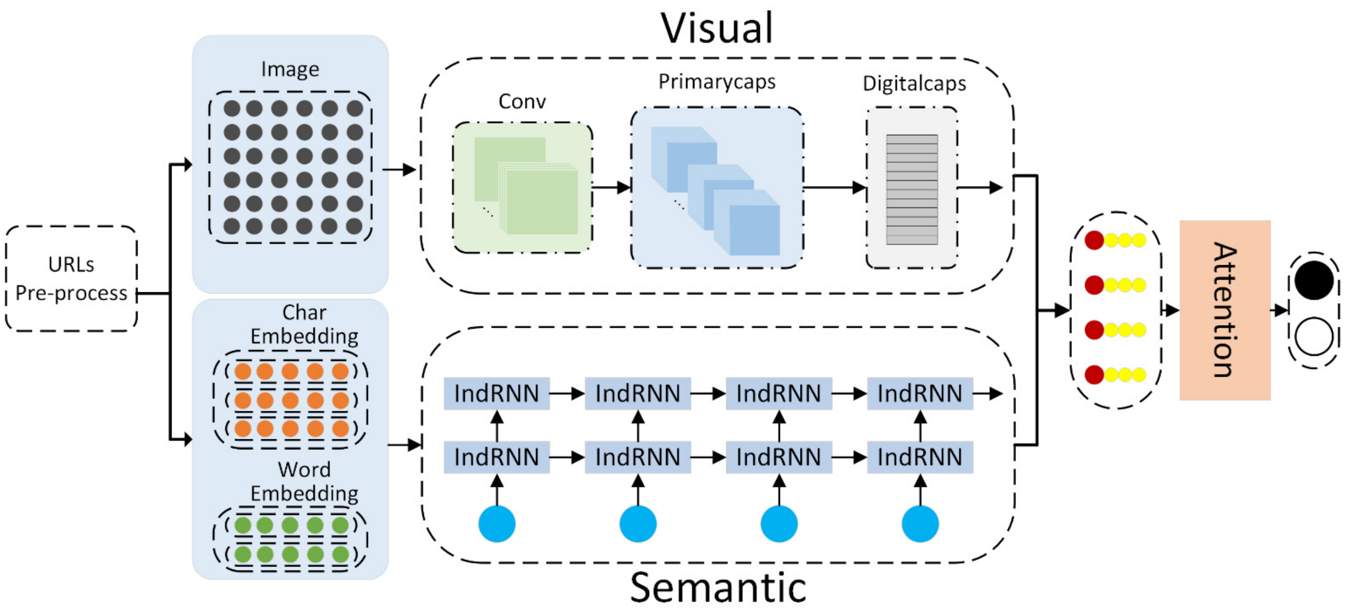


FIGURE 1. Overall framework of model.

Hence, the embedding vector is optimized for each character and word leading to the semantically similar embedded vector of letters or words being closer to each other [29] and fully expressing the meaning of the character and the association between the characters. Especially, through this matrix, a URL is converted to a two-dimensional tensor.

2) VISUAL FEATURE

Vasan *et al.* [30] suggested that pe files of the same family have similar texture structures. They converted the malware into grayscale images and used it for classifying against the CNN network. Within malicious URLs, the URL of the same family or the URL generated by the same tool possesses a similar structure. Therefore, the malicious URL is also converted into images. This method can intuitively represent the spatial pattern and structure of URLs, moreover, it is also easier to identify the spatial similarity of URLs generated by the same family or the same tool.

Fig. 2 shows the process of converting the URL to a grayscale image. First, each character in the given URL is converted to a decimal integer (ASCII), to obtain a new decimal vector representing the malware sample. Then, the decimal vector is reshaped to a two-dimensional matrix and visualized as a grayscale image. Therefore, the entire URL can be represented by a grayscale image as an effective input and very fast operation for the subsequent model.

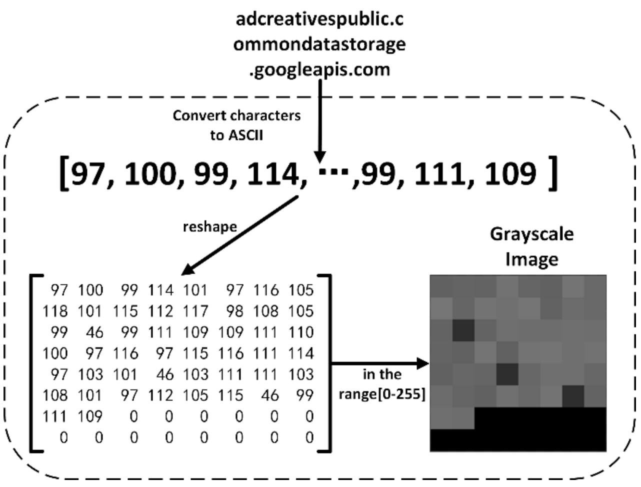


FIGURE 2. Visualization process.

B. MODEL

In this part, we introduce our model including three parts, namely IndRNN, CapsNet, and Attention in total, for which the overall flow is represented in Fig. 1. The embedded vector sequence was utilized as the input of the IndRNN and the texture image was used as the input of the CapsNet. Moreover, the outputs of the two networks were entered into an attention mechanism to extract useful valid features for classification. Ultimately, the sigmoid classifier was used to calculate the class probabilities based on the final feature.

1) INDEPENDENT RECURRENT NEURAL NETWORK

RNN has been extensively used in sequence learning problems such as action recognition and language processing to achieve remarkable results. However, it is actually difficult to build and train a deep RNN since using hyperbolic tangent and sigmoid activation functions in LSTM and GRU will lead to gradient decay over layers. An IndRNN model was proposed in the literature [15] to avoid the gradient exploding problem of traditional RNN and solve the problem of gradient vanish in multi-layer LSTM and multi-layer GRU

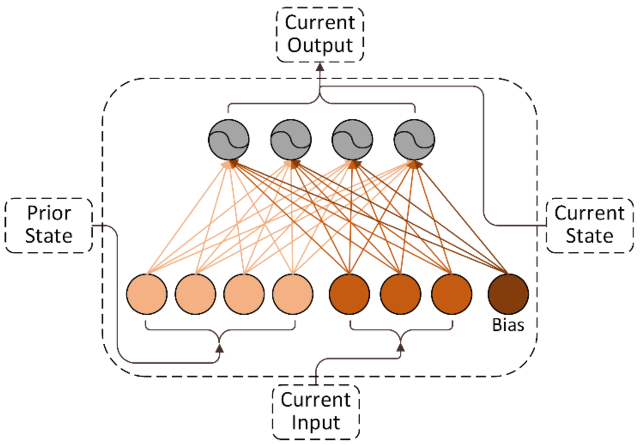


FIGURE 3. **The unit structure in IndRNN.**

transmission. To better learn the information from the characters embedded vector sequence, the IndRNN is selected. The IndRNN unit structure is shown in Fig. 3.

Considering an input, $S = \{s_1, s_2, \ldots, s_t\}$, s_t represents the embedding vector of the t-th character in a URL. The hidden status of IndRNN is updated as:

$$h_t = \sigma(Ws_t + U \odot h_{t-1} + b_1) \tag{1}$$

where $W \in R^{N \times M}$ and $U \in R^N$ are the input weight and the recurrent weight denotes the Hadamard product. In IndRNN, each neuron only receives input from the current time and hidden state at the previous time, and each neuron independently processes one type of spatiotemporal pattern. Normally, traditional RNN is considered as a multi-layer perceptron model sharing parameters through time, however, IndRNN represents a new perspective of independently gathering spatial patterns (through w) over time (through u). Each neuron in each layer independently processes the output of all neurons in the previous layer, hence, the difficulty of constructing a deep network structure is reduced while enhancing the ability to model longer sequences. Moreover, using ReLU activation functions, IndRNN is more robust after training.

2) CAPSULE NETWORK

In 2011, Hinton *et al.* [31] first introduced the capsule network. The core idea is to use capsules to replace neurons in RNN, hence, the network can retain spatial relationships and detailed pose information between objects. Compared to CNNs, the pooling layer is removed by CapsNet causing feature loss, and the spatial relationship is fully utilized between each feature in the image to obtain the positional relationship between high-level features and low-level features as a kind of classification feature. Sabour *et al.* [14] further proposed a dynamic routing algorithm between capsules and a capsule neural network structure. CapsNet uses vector capsules to replace neurons in CNNs, dynamic routing to replace pooling operations, and Squash functions to replace ReLU activation functions. Moreover, they indicated that the multi-layer capsule system is effective in image recognition tasks and is

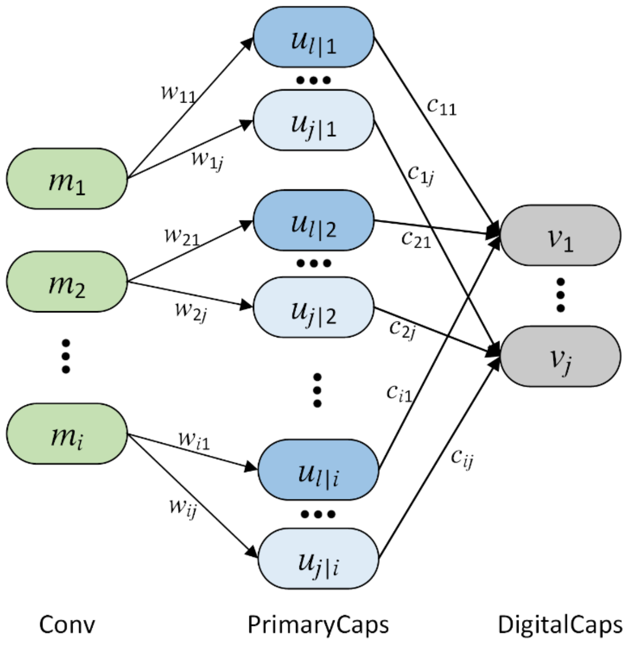


FIGURE 4. **The CapsNet structure.**

significantly better than convolutional networks in recognizing highly overlapping objects.

The structure of the capsule network utilized in our work is shown in Fig. 4. First, to process URL image a simple convolutional layer is used:

$$m_l = f(W_1 \circ X_{l:l+k-1} + b_1) \tag{2}$$

$$\mathbf{M}_{k1} = \left[m_1, m_2, \cdots, m_{(L-N+1)}\right] \tag{3}$$

where f denotes the ReLU activation function, $W_1 \in R^{N \times d}$ is the convolution filter, and b_1 is the bias. By sliding the filter on the image, the extracted local features are stitched together into a feature map.

Then, the primary capsule layer reaches encapsulating the features at the same position in the feature map into a corresponding capsule:

$$p_i = g(W_2 M^i + b_2) \tag{4}$$

$W_2 \in R^{k \times 1 \times l}$ is the transformation matrix, where k denotes the dimension of the capsule. M^i is the i-th row vector of the feature map, b_2 shows the bias, and g is the squash function:

$$g(x) = \text{squash}(x) = \frac{\|\mathrm{x}\|^2}{1 + \|x\|^2} \frac{\mathrm{x}}{\|x\|} \tag{5}$$

The digital capsule layer is the final layer, each capsule is obtained by :

$$V_{\mathrm{j}} = g(\sum_{i=1} c_i \hat{u_i}) \tag{6}$$

where c_i is the coupling coefficient updated by the dynamic routing algorithm. $\hat{u_i}$ denotes the prediction vector of the current output of the previous layer obtained by the linear transformation matrix W_3:

$$\hat{u}_i = W_3 p_i \tag{7}$$

3) ATTENTION

The attention mechanism in deep learning is based on the attention of human thinking. Limited attention resources are used by humans to quickly filter out high-value information from a huge deal of information greatly improving the accuracy and efficiency of information processing. The main task of the attention mechanism in deep learning is to choose the more critical information for the current task target from numerous input information by weighting the target data. To combine the extracted visual and semantic features, not all parts contribute equally to the ultimate classification. To improve the representational ability of key features, we used the attention mechanism to fuse the networks' outputs and model multimodal scores. First, the semantic (S) and visual vectors (V) are inserted into the one-layer MLP to get:

$$u_{Sem,i} = \tanh\left(W_{Sem} S_j + b_{Sem}\right) \tag{8}$$

$$u_{Vis,i} = \tanh\left(W_{Vis} V_j + b_{Vis}\right) \tag{9}$$

$$u_{Con} = [u_{Sem}, u_{Vis}] \tag{10}$$

where u_i denotes the i-th hidden representation and "[]" represents the concatenation operation. Hence, the weight is determined:

$$a_i = \exp\left(u_i^T u_{con}\right) / \sum_i \exp\left(u_i^T u_{con}\right) \tag{11}$$

After that, the weighted sum of multimodal vectors is calculated based on weights:

$$V_{Con} = \sum_i a_i \left[S_i, V_i\right] \tag{12}$$

The final vector V can be utilized as the input for the final classification. Moreover, the sigmoid function is used to compute the class probability of each label:

$$y = Sigmoid(v) = 1/(1 + \exp(-v)) \tag{13}$$

Ultimately, URL classification is performed based on the threshold we set:

$$y_{\text{binary}} = \begin{cases} 0, & y \leq \text{threshold} \\ 1, & y \geq \text{threshold} \end{cases} \tag{14}$$

IV. EXPERIMENTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT

The experimental environment and configuration information are as follows:

Computer configuration: Windows Server 2012 R2, 256GB of memory, CPU Intel(R) Xeon(R) gold 5117 @ 2.00GHz, GPU NVIDIA Tesla V100-PCIE-16GB, 256GB SSD + 30T HDD of hard disk;

Python version:3.6; Deep Learning Library: Keras, Tensorflow.

B. DATASET

The dataset includes benign and malicious instances: The malicious dataset was collected from a well-known free anti-phishing website PhishTank and a website malwaredomain-list collecting a blacklist of malicious websites. The benign dataset was collected through the Alexa website ranking and web search. We acquired a total of 66,017 URLs, of which 32,519 were benign and 33,498 were malicious.

C. EVALUATION METRICS

For all experiments, the 5 fold cross-validation technique was adopted and unified performance indicators such as accuracy (15), precision (16), recall (17), and F (18) were used to assess the performance of the model.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$P = \frac{TP}{TP + FP} \tag{16}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{17}$$

$$\mathrm{F} = \frac{2 \times P \times R}{P + R} \tag{18}$$

where *TP* indicates the number of samples correctly classified as malicious, *TN* denotes the number of samples correctly classified as benign, *FP* shows the number of samples wrongly classified as malicious, and *FN* indicates the number of samples wrongly classified as benign.

D. THE EFFECT OF MODEL PARAMETERS ON EXPERIMENTAL RESULTS

The setting of experimental hyperparameters has great importance to the overall performance of the model. Rational hyperparameter settings can utilize features more effectively and enhance the accuracy of the ultimate URLs detection. Thus, to optimize the experimental results, various hyperparameters were tested on the same dataset to define the optimal feature vector dimension and number of IdnRNN network layers. Moreover, the influence of different data sizes was also measured. To avoid the influence of other factors on the experimental results, each experiment is carried out when other parameters are fixed or even optimal.

In order to determine the optimal dimension of the feature vector, the dimensions of visual features and semantic features were combined from low to high for experiments. According to Table 1, the F-values are the two highest and are very close when the total dimensions are 105 and 185. However, the recall rate reached 99.98% when the total dimension is 185, which is better than the former. Although the accuracy of the latter is not as high as that of the former, it is even more intolerable in URL detection to remove a malicious URL. Comprehensively, 185 is used as the optimal parameter of the feature dimension.

When other parameters were fixed to the optimal, the effect of different IndRNN layers on the model was explored through experiments. According to Fig. 5, when the number

TABLE 1. Experimental results of feature dimensions.

Total	Character + Lexica	Texture	Acc(%)	P(%)	R(%)	F(%)
105	75	30	99.78	99.61	99.93	99.77
125	85	40	99.67	99.35	99.98	99.66
145	95	50	99.75	99.52	99.98	99.75
165	105	60	99.65	99.38	99.92	99.65
185	115	70	99.78	99.57	99.98	99.78
205	125	80	99.7	99.51	99.87	99.69

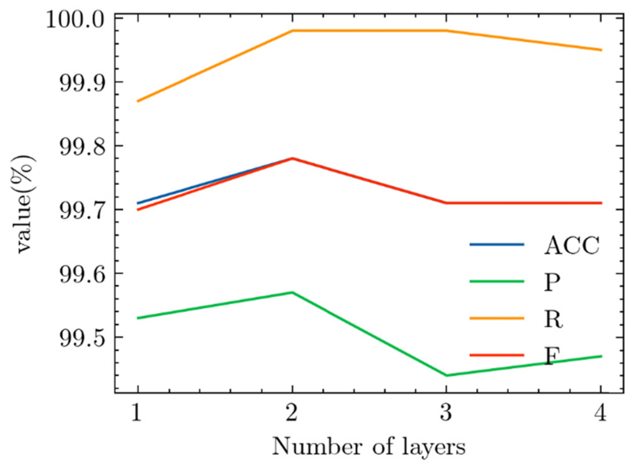


FIGURE 5. The experimental result of the IndRNN layers.

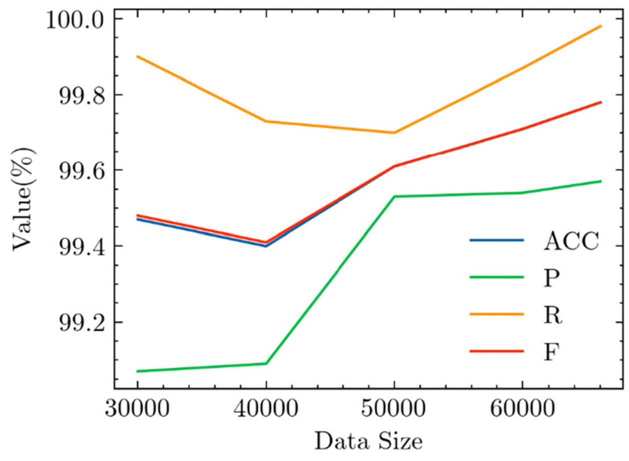


FIGURE 6. The experimental result of data size.

of layers is 2, the values of all indicators are greater than those of other groups. Thus, the 2-layer IndRNN is more conducive to extracting the semantic features of an embedded vector sequence.

To verify the model's reliability, we tested the effect of the size of data on classification. Based on Fig. 6, when the data size is only 30,000, the accuracy rate of 99.47% and other good index values can be achieved. When the data volume is 40,000, other indicators are reduced except for precision. The possible reason is that the distribution of the newly added data is inconsistent with the original data set, leading to an increase in output entropy. By adding the data continuously, the distribution between the data sets tends to be consistent, and the ultimate accuracy rate reaches 99.78%.

TABLE 2. Necessity of model components.

Components	Acc(%)	P(%)	R(%)	F(%)
IndRNN	99.52	99.22	99.81	99.51
CapsNet	99.67	99.47	99.86	99.66
Attention	98.95	98.71	99.16	98.93
Attention-based IndRNN	99.59	99.78	99.37	99.57
Attention-based CapsNet	99.71	99.57	99.86	99.71
IndRNN+CapsNet	99.68	99.5	99.86	99.68
Our	99.78	99.57	99.98	99.78

E. THE NECESSITY OF MODEL COMPONENTS

To verify whether each component is redundant, we designed such a set of experiments: the same training was performed by combining different components to generate a new model. As shown in Table 2, the corresponding network and sigmoid classifier were used by the first three experiments directly. To verify the effectiveness of texture features and CapsNets, only one branch of IndRNN was used by the Attention-based IndRNN model to learn semantic features. Attention-based CapsNet was also a model designed to verify the effectiveness of text features and IndRNN. The features learned by the two branches were merged by IndRNN+CapsNet and sent directly to the Sigmoid classifier.

According to the experimental results in Table 2, the three single models also have excellent performance, among which CapsNet is the best. This also indicates that texture features are effective for URL detection. Compared to our method, due to no visual information branch in Attention-based IndRNN, Attention-based CapsNet has no semantic information branch and IndRNN+CapsNet possesses no attention mechanism, therefore, they yield relatively weak experimental results. This also proves that texture features and CapsNet, text features and IndRnn, and attention mechanism all have a role in our model. As we expected, our model integrates the advantages of each component indicated by the final accuracy rate of 99.78% and the F-value of 99.78%. Multi-modal knowledge can be learned from URL data since semantic embeddings and visual vectors are shared across the network. Furthermore, the fusion of the two also results in improved performance.

F. COMPARISON WITH METHODS

Comparative experiments with four new methods based on different models were also conducted. We implemented the model mentioned below and performed experiments on our dataset. As shown in Table 3, all methods had a good performance. Wei *et al.* [32] also utilized character embedding technology. They utilized the character embedding sequence of the URL as the input of the CNN network to obtain

TABLE 3. Effect of different algorithm models on experimental results.

models	Acc(%)	P(%)	R(%)	F(%)
CNN [32]	99.56	99.75	99.35	99.55
LSTM [12]	99.55	99.78	99.29	99.53
CapsNet[33]	99.72	99.50	99.90	99.70
Bi-LSTM [34]	99.68	99.76	99.59	99.68
Bi-IndRNN[35]	99.68	99.43	99.93	99.68
CNN+LSTM [36]	99.68	99.43	99.92	99.67
Our	99.78	99.57	99.98	99.78

a reasonable accuracy. The model of Bahnsen *et al.* [12] combined character embedding and LSTM network indicating almost the same performance as CNN. Furthermore, since CapsNet can learn features that are different from convolutional networks, a single CapsNet can perform better with an accuracy of 99.72%. In Liang's method [34], the Bi-LSTM network-based model had a performance improvement. In [35]'s method, through using the Bi-IndRNN model to learn the host features and URL information features, and finally a recall rate of 99.93% is obtained. The proposed model [36] is somewhat similar to our model, and its performance was similar to that of Bi-LSTM. It is observed that our model has some improvements over the previous model, moreover, the results in the table are also encouraging, with an accuracy rate of 99.78%.

G. DISCUSSION

Considering the above experimental results, we found that our model achieves the best performance when the feature dimension is 185 and the number of IndRNN layers is 2. By comparing them with the other combinations of IndRNN, CapsNet and Attention in TABLE 2, it can be summed up that the model proposed in this paper combines their advantages and outdoes them in a variety of evaluation indicators. And it also shows that it is feasible to obtain the semantic information and visual information of URL at the same time. Converting the URL to an image and the sparse vector into a dense real number vector are simple and effective. The experimental results in TABLE 3 also show that the CapsNet is better than CNN in learning the texture feature of the image, and IndRNN has better robustness than LSTM in sequence learning problems. This proves that we are right to choose IndRNN and CapsNet. Furthermore, the accuracy and recall rates have reached 99.78% and 99.98%, the combination of the two leads to achieving performance beyond traditional models.

V. CONCLUSION

This study explored the possibility of distinguishing legitimate URLs from malicious URLs by combining the two technologies of CapsNet and IndRNN. To evaluate the proposed method, a dataset was used including over 30,000 malicious URLs from the PhishTank and the malwaredomainlist and more than 30,000 legitimate URLs ranked by Alexa. The proposed method gave a high classification accuracy of 99.78%. Hence, word embedding technology can be used to embed sufficient semantic knowledge in malicious URLs into distributed vectors, then combine with gray images and use neural network models for classification. It only requires some simple processing of the original URL and does not rely on any other complex or expert features. Compared to other methods, our method has a better detection effect.

Although the proposed model performs well, further improvement is still required along with further studies to improve the entire system. We will try to modify our model structure based on malicious classes to perform multiple classifications. In future work, newer better-performing versions may be selected to replace some of these components.

REFERENCES

[1] Microsoft, "Microsoft security intelligence repor volume 24: January-December 2018," Microsoft Corp., Redmond, WA, USA, Tech. Rep., Jan. 2018.

[2] M.-Y. Kan, "Web page classification without the Web page," in *Proc. 13th Int. World Wide Web Conf. Alternate Track Papers Posters*, 2004, pp. 262–263.

[3] 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. (IMC)*, 2010, pp. 48–61.

[4] 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 *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 1990–1994.

[5] L. Bilge, "EXPOSURE: Finding malicious domains using passive DNS analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2011.

[6] M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, "Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2018, pp. 295–314.

[7] Y. Li, Z. Yang, X. Chen, H. Yuan, and W. Liu, "A stacking model using URL and HTML features for phishing webpage detection," *Future Gener. Comput. Syst.*, vol. 94, pp. 27–39, May 2019.

[8] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, "Malicious Web content detection by machine learning," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 55–60, Jan. 2010.

[9] Y. Fang, C. Huang, L. Liu, and M. Xue, "Research on malicious JavaScript detection technology based on LSTM," *IEEE Access*, vol. 6, pp. 59118–59125, 2018.

[10] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, "Detection of phishing webpages based on visual similarity," in *Proc. Special Interest Tracks Posters, 14th Int. Conf. World Wide Web*, 2005, pp. 1060–1061.

[11] F. C. Dalgic, A. S. Bozkir, and M. Aydos, "Phish-IRIS: A new approach for vision based brand prediction of phishing Web pages via compact visual descriptors," in *Proc. 2nd Int. Symp. Multidisciplinary Stud. Innov. Technol. (ISMSIT)*, Oct. 2018, pp. 1–8.

[12] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonzalez, "Classifying phishing URLs using recurrent neural networks," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, Apr. 2017, pp. 1–8.

[13] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Secur.*, 2011, p. 4.

[14] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3856–3866.

[15] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (IndRNN): Building a longer and deeper RNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5457–5466.

[16] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, 2014, pp. 1–21.

[17] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 346–350.

[18] B. Sun, M. Akiyama, T. Yagi, M. Hatada, and T. Mori, "Automating URL blacklist generation with similarity search approach," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 4, pp. 873–882, 2016.

[19] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy "SpyProxy: Execution-based detection of malicious Web content," in *Proc. 16th USENIX Secur. Symp., USENIX Secur. Symp. (SS)*, 2007, p. 3.

[20] C. Liang, "A redox-active covalent organic framework for the efficient detection and removal of hydrazine," *J. Hazardous Mater.*, vol. 381, Jan. 2020, Art. no. 120983.

[21] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing Web sites," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007, pp. 639–648.

[22] F. Vanhoenshoven, G. Napoles, R. Falcon, K. Vanhoof, and M. Koppen, "Detecting malicious URLs using machine learning techniques," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.

[23] N. S. Gawale and N. N. Patil, "Implementation of a system to detect malicious URLs for Twitter users," in *Proc. Int. Conf. Pervas. Comput. (ICPC)*, Jan. 2015, pp. 1–5.

[24] A. Y. Daeef, R. B. Ahmad, Y. Yacob, and N. Y. Phing, "Wide scope and fast websites phishing detection using URLs lexical features," in *Proc. 3rd Int. Conf. Electron. Design (ICED)*, Aug. 2016, pp. 410–415.

[25] N. Jayakanthan, A. V. Ramani, and M. Ravichandran, "Two phase classification model to detect malicious URLs," *Int. J. Appl. Eng. Res.*, vol. 12, no. 9, pp. 1893–1898, 2017.

[26] N. Azeez, B. Salaudeen, S. Misra, R. Damasevicius, and R. Maskeliunas, "Identifying phishing attacks in communication networks using URL consistency features," *Int. J. Electron. Secur. Digit. Forensics*, vol. 12, no. 2, p. 200, 2020, doi: 10.1504/IJESDF.2020.106318.

[27] J. Saxe and K. Berlin, "EXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," 2017, *arXiv:1702.08568*. [Online]. Available: http://arxiv.org/abs/1702.08568

[28] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, *arXiv:1611.00791*. [Online]. Available: http://arxiv.org/abs/1611.00791

[29] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1746–1751.

[30] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of CNN architectures (IMCEC)," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101748, doi: 10.1016/j.cose.2020.101748.

[31] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. 21th Int. Conf. Artif. Neural Netw. (ICANN)*, 2011, pp. 44–51.

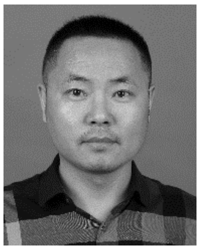
[32] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Wozniak, "Accurate and fast URL phishing detector: A convolutional neural network approach," *Comput. Netw.*, vol. 178, Apr. 2020, Art. no. 107275.

[33] Y. Huang, J. Qin, and W. Wen, "Phishing URL detection via capsule-based neural network," in *Proc. IEEE 13th Int. Conf. Anti-Counterfeiting, Secur., Identificat. (ASID)*, Oct. 2019, pp. 22–26.

[34] Y. Liang, J. Deng, and B. Cui, "Bidirectional LSTM: An innovative approach for phishing URL identification," in *Proc. Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, 2019, pp. 326–337.

[35] W. Huanhuan *et al.*, "An algorithm based on bi-IndRNN for analysis and detection of malicious URL," *J. Xinjiang Univ., Natural Sci. Ed.*, vol. 36, no. 154, pp. 174–181, 2019.

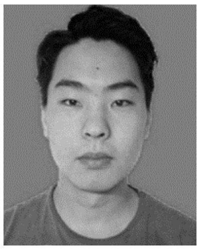
[36] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing URL detection via CNN and attention-based hierarchical RNN," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 112–119.



JIANTING YUAN received the M.S. degree from Xidian University, in 2008. His research direction was cryptographic protocol. His research interests include cyber security, critical information infrastructure protection, personal privacy protection, and data security.

GUANXIN CHEN was born in Nanyang, Henan, China, in 1996. He is currently pursuing the M.S. degree with the School of Computer Science and Engineering, Xinjiang University, Ürümqi, China. His research interests include cyber security and deep learning

SHENGWEI TIAN received the Ph.D. degree in computer science and technology from Xinjiang University, in 2010. He is currently a Professor with the Xinjiang University of Technology. His research interests include intelligence computing, image processing, and natural language processing.



XINJUN PEI was born in Tacheng, Xinjiang, China, in 1995. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, Changsha, China. Since 2017, he has been engaged in the direction of information security. His research interests include deep learning, edge computing, and the IoT security.

• • •