Hindawi
Security and Communication Networks
Volume 2019, Article ID 2595794, 15 pages
https://doi.org/10.1155/2019/2595794

Research Article

PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks

Weiping Wang,[1] Feng Zhang,[1] Xi Luo,[2] and Shigeng Zhang[1,3]

[1]*School of Computer Science and Engineering, Central South University, Changsha, China*
[2]*Hunan Provincial Key Laboratory of Network Investigational Technology and Department of Information Technology, Hunan Police Academy, Changsha, China*
[3]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*

Correspondence should be addressed to Xi Luo; 9075111@qq.com and Shigeng Zhang; sgzhang@csu.edu.cn

Received 7 May 2019; Accepted 29 August 2019; Published 29 October 2019

Academic Editor: Angel M. Del Rey

Through well-designed counterfeit websites, phishing induces online users to visit forged web pages to obtain their private sensitive information, e.g., account number and password. Existing antiphishing approaches are mostly based on page-related features, which require to crawl content of web pages as well as accessing third-party search engines or DNS services. This not only leads to their low efficiency in detecting phishing but also makes them rely on network environment and third-party services heavily. In this paper, we propose a fast phishing website detection approach called PDRCNN that relies only on the URL of the website. PDRCNN neither needs to retrieve content of the target website nor uses any third-party services as previous approaches do. It encodes the information of an URL into a two-dimensional tensor and feeds the tensor into a novelly designed deep learning neural network to classify the original URL. We first use a bidirectional LSTM network to extract global features of the constructed tensor and give all string information to each character in the URL. After that, we use a CNN to automatically judge which characters play key roles in phishing detection, capture the key components of the URL, and compress the extracted features into a fixed length vector space. By combining the two types of networks, PDRCNN achieves better performance than just using either one of them. We built a dataset containing nearly 500,000 URLs which are obtained through Alexa and PhishTank. Experimental results show that PDRCNN achieves a detection accuracy of 97% and an AUC value of 99%, which is much better than state-of-the-art approaches. Furthermore, the recognition process is very fast: on the trained PDRCNN model, the average per URL detection time only cost 0.4 ms.

1. Introduction

With the rapid development of the Internet in the past decade, some attackers have forged phishing websites to imitate real enterprise websites in order to induce normal users to disclose personal information, e.g., bank accounts, mail accounts, and passwords. This kind of phishing attacks is now very common and growing rapidly. In the report recently released by the Antiphishing Working Group (APWG) [1], it mentions that APWG members have been detecting more than 250,000 phishing attacks using 195,475 different domains from 2015 to 2016. Both numbers are the highest record since APWG began reporting phishing statistics in 2007.

Phishing detection has received much research attention in recent years. Existing phishing detection approaches mainly falls into three different categories: approaches based on black- and whitelist, approaches based on web page visual similarity, and approaches based on URL and website content features.

The black- and whitelist-based approaches detect whether a given URL is phishing by matching it with a list of known phishing sites that have been identified by the third party. Such approaches are usually used in industrial engineering to

intercept URLs [2, 3] located in a given list. The limitation of this method is that on the one hand, they rely on the detection results provided by third parties like Google Safe Browsing API, which has a certain lag and cannot defend against 0-day phishing attacks, and on the other hand, not all the whitelisted pages are irrationally labeled as suspicious, which is unfair to most benign sites.

The visual similarity-based method is to extract visual features from phishing websites, and then use these features to identify phishing web pages. The disadvantages of these methods are that they need to retrieve the visual content of the web page, and any distortion of the web page content will lead to misclassification. And the extraction and matching process of visual features will consume computational resources [4–7].

The method of distinguishing between phishing and benign pages based on URL and web content features is the most important method for phishing detection. Such methods need to obtain relevant information of URL corresponding pages, such as obtaining page keywords and page forms, and always need relevant features, such as ranking and IP, of the page are obtained by means of a search engine service or a DNS service.

Most current phishing detection approaches exploit the URL and web content features to distinguish between phishing and benign pages [8–13]. These approaches find features that are different in phishing benign pages, and use experimental heuristics to detect phishing pages. They need the information which is related to page content of the URL, including page keywords and page forms. Moreover, they also need relevant features such as ranking of the target website and its IP address, which need to access third-party services such as search engine and DNS.

Machine learning techniques have also been integrated with this kind of approaches to improve detection performance [14–23]. These phishing website features identified through artificial feature engineering can effectively transfer the knowledge of security experts to computers and turn security issues into computational problems. Then, through feature extraction and sample training, it has achieved good detection results. These methods, based on URL and web content features, require not only local computing resources but also network access and third-party services. The detection efficiency is low, and when phishing attacks continue to change and escalate, the effectiveness of these features is waning.

In this paper, we propose a new phishing website detection method PDRCNN, which only uses the URL to detect phishing and does not need third-party services such as search engine or DNS services. PDRCNN extracts the structural and semantic features in the phishing website URL through the deep learning model for the detection of phishing website. Our approach is independent to external information bases and is very fast with detection time less than 0.4 ms per URL. To our knowledge, PDRCNN is the first that can perform precise and fast phishing detection only with URL information. Our main contributions are summarized as follows:

(i) We first proposed a phishing detection model with deep learning, and it can detect phishing sites quickly and accurately not relying on third-party data and search engine results.

(ii) We combine the advantages of RNN and CNN in processing text data. At first, we use the RNN to extract the global features from the URL, and then use the CNN to extract the local features.

(iii) We build a large-scale data set through Alexa and PhishTank websites, which contains nearly 500,000 experimental samples. The accuracy of the experimental results reached 97%, and the AUC value reached 99%.

(iv) We design four baseline models, and the experimental results indicate that PDRCNN can better detect phishing website URLs than existing machine learning-based methods and general n-gram methods.

The remainder of this paper is structured as follows: The second section reviews related works. The third section introduces the basic idea of PDRCNN. The fourth section introduces the design of the PDRCNN method in detail. The fifth section describes the experimental results and the analysis of the experimental results, while the summary of this article discussed in Section 6.

2. Related Works

2.1. Blacklist/Whitelist-Based Approaches. The blacklist- and whitelist-based detection method needs to maintain a list of information of a known phishing website in order to check the currently visited. This list, which needs to be constantly updated, contains information such as known phishing URLs, IP addresses, and domain names. It determines whether a website is a phishing page by verifying whether it is in a black or whitelist.

Google Safe Browsing API [2] is an interface provided by Google to query whether a given URL address exists on Google's phishing website blacklist. In 2008, Han et al. [3] proposed a whitelist-based phishing website detection method that records the LUI (login user interface) information and IP address of each URL accessed by the user. When a user visits a website included in the whitelist and submits account information, an alarm will be generated, if the website information does not match the information of the white list. The disadvantage of this method is that it will alert the user when visiting the normal website for the first time.

2.2. Visual Similarity-Based Approaches. The detection method based on page visual similarity needs to take a snapshot of the web page, requires large calculation and storage resources, and mainly detects the phishing website with similar page visuality. Liu et al. [4] proposed a method for judging the website type by comparing the visual similarity between phishing websites and nonphishing websites. The method utilizes the HTML DOM tree to segment the

page based on "visual cues" and then uses three evaluation metrics to assess the visual similarity between the site to be tested and the legitimate site: block level similarity, layout similarity, and overall style similarity of web pages. The method can detect phishing with a low false detection rate, but it is time-consuming. Moreover, it depends largely on the results of web page segmentation. Different from this, Mao et al. [5] proposed a method to detect phishing websites by detecting the key element similarity method related to CSS files.

Shekokar et al. [6] proposed a detection method based on the URL and web page similarity. They proposed the LinkGuard algorithm to determine whether a URL is suspicious and used an image-based page matching approach to obtain similarity between the target pages and pages in phishing websits. Then, a threshold is used to detect whether the target page is a phishing page. Chiew et al. [7] proposed Phishdentity that uses favicon extracted from the website and uses Google as the image search engine to discover potential phishing attempts. Phishdentity does not require intensive analysis of text-based or image-based content, and thus increases detection speed.

2.3. Heuristic-Based Approaches. The heuristic detection methods [8–13] are based on the similarity between phishing pages, the statistical characteristics of phishing, or the prior knowledge of experts. It extracts multiple features from the detected phishing pages and generalizes them into a set of heuristic features. Phishing attack detection is then implemented based on these characteristics.

Zhang et al. [8] proposed a heuristic-based phishing detection method named CANTINA. It uses a Google search engine to retrieve keywords and domain names in a web page and determines whether the page is legitimate based on the results returned by the search and other heuristic features. Prakash et al. [9] proposed PhishNet, which enumerates the simple phishing website URL based on five heuristic rules. Shahriar and Zulkernine [10] tested the credibility of suspicious websites to determine whether the site was a phishing site. They proposed a finite state machine (FSM) method that tracks web page forms and corresponding responses to evaluate web page behavior. Ramesh et al. [11] proposed a method to detect phishing websites by reviewing web pages and determining all direct or indirect links related to the web pages. The method achieves high detection accuracy, but it is time-consuming because it relies on search engines and third-party services such as the DNS query. Jain and Gupta [13] proposed the phishing detection algorithm (PDA) to determine whether a suspicious URL is a phishing website. PDA mainly determines whether a URL is legal by calculating the number of hyperlinks in the suspicious web page. The paper gives a result of testing true positive of 86.07% and false negative of 1.48% on 1120 phishing pages (from PhishTank) and 405 legal pages.

2.4. Machine Learning-Based Approaches. The phishing detection, based on machine learning, regards the phishing detection problem as a text classification or clustering problem and uses various classification and clustering algorithms (e.g., K-nearest neighbor, C4.5, support vector machine, and random forest) to detect phishing attacks.

Aburrous et al. [14] proposed a system to detect phishing pages in e-banking. They applied 27 features to assess the risk of phishing attacks on e-banking pages. Xiang et al. [15] proposed CANTINA+, an improved version of CANTINA. This method contains three stages: First, it uses HTML DOM, search engine, and third-party services to extract eight novel features that reveal the characteristics of phishing attacks. Second, it uses the heuristic rules to filter out pages that do not have a login box before performing the classification process. Third, it selects 15 highly expressing phishing features and uses machine learning algorithms to perform phishing page detection. He et al. [16] proposed a system based on page content, HTTP transactions, and search engine results. They use the SVM algorithm to identify phishing pages and achieve a detection accuracy of 0.97. Mohammad et al. [17] proposed a model based on conventional features and summarized the prediction error rate generated by a set of association classification (AC) algorithms. Abdelhamid et al. [18] used the multilabel classifier-based classification algorithm (MCAC) to extract its rules from the training data.

Zhang et al. [19] proposed a new model with five novel features and a sequence minimum optimization (SMO) algorithm for classifying and detecting Chinese phishing websites. Moghimi et al. [20] proposed a phish detector, which first uses SVM to train the phishing detection model, and then uses the SVM_DT to extract the hidden phishing. The proposed approach achieves true positive of 0.99 and false negative of 0.001 in a large dataset. However, this method assumes that the pages of the phishing website only use the content of the legitimate page, which does not hold in practice. Shirazi et al. [21] proposed a method that relies on only domain name based features for detection of phishing websites. Babagoli et al. [22] proposed a phishing website detection method that utilizes a metaheuristic-based nonlinear regression algorithm together with a feature selection approach. Recently, Chiew et al. [23] proposed a feature selection framework for machine learning-based phishing detection system called hybrid ensemble feature selection (HEFS). HEFS uses a cumulative distribution function gradient (CDF-g) algorithm to produce primary feature subsets and uses data perturbation ensemble to yield secondary feature subsets. HEFS performs well when it is integrated with the Random Forest classifier.

2.5. Deep Learning-Based Approaches. The method of detecting phishing websites based on the deep learning model is to design a reasonable deep learning model, construct the input required by the model, and extract the features through the deep learning model to complete the detection of the phishing website URL. In this type of approaches, the selection and the construction of the model

input will directly affect whether the model is effective. Currently, the commonly used models for detecting phishing websites are CNN and RNN.

Correa Bahnsen et al. [24] proposed using the LSTM model to detect phishing URLs. This method first encodes the URL string using the one-hot encoding method, and then inputs each encoded character vector into the LSTM neurons for training and testing. The method achieved an accuracy of 0.935 on the Common Crawl and PhishTank datasets. Chen et al. [25] also proposed an LSTM-based phishing page detection approach. Nivaashini and Soundariya [26] proposed to use the autoencoder to extract the representation of the phishing website URL. It requires third-party services such as PageRank and DNS query. Hung et al. [27] proposed the URLNet method for malicious website URL detection. They extract char-level and word-level features based on URL strings and use CNN network for training and testing.

2.6. Summary of Existing Approaches. We survey most current existing phishing detection approaches in Table 1. We mainly focus on four different aspects of these phishing detection approaches: (1) the approach's dependence on the search engine, (2) the approach's dependence on third-party organizations' data; (3) whether the approach depends on a specific language, and (4) the number of benign samples and phishing samples used to evaluate the approach. From the table, we can find that most existing approaches are based on page-related features. The acquisition of these features requires crawling web pages and accessing third-party search engine services or DNS services. This causes inefficient detection of phishing websites and relies heavily on the network environment and the third-party services.

3. Overview of PDRCNN

3.1. Motivation. Although the URL itself has already been used as a feature in existing phishing website identification approaches [15, 28–30], e.g., the length of the URL, whether the URL contains a nested domain name, and whether a special character such as "@" or "-" appears in the URL, it is however generally believed that the accuracy of recognition by relying solely on URL features and machine learning methods is not high. Table 2 shows the list of nine artificial phishing website character features.

We use statistical knowledge to perform statistics on these 9 URL character-level features, as shown in Figure 1. The yellow bar indicates that the corresponding feature in the normal website URL data is "1." The height of blue bars indicates the number of corresponding features in the phishing website URL data of "1."

In Figure 1, we can clearly see that the phishing website URL and the normal website URL have significant differences in these 9 features. Among the features 3, 4, 5, 7, 8, and 9, the number of phishing website feature values is significantly larger than the benign website.

We are also concerned that some research supports a certain correlation between phishing website URLs. In 2010, Prakash et al. [9] proposed that phishing website attackers would build a new phishing website by modifying a part of the URL on the basis of the existing phishing website URL. In other words, the phishing website URLs generated by the same phishing attacker or phishing attack organization are similar in structure or semantics. PhishNet proposes to divide the known phishing URL into five parts: domain, top-level domain, directory, file name, and query string, i.e., http://domain.TLD/directory/filename?query_string. Some new phishing website URLs can be exhaustively combined according to certain rules.

For example, two phishing URLs, http://www.xyz.com/online/signin/ebay.htm and http://www.abc.com/online/signin/paypal.htm, are known to combine new phishing URLs, http://www.xyz.com/Online/signin/paypal.htm and http://www.abc.com/online/signin/ebay.htm. This finding indicates that there is a certain correlation between the texts contained in the phishing URL.

At the same time, deep learning has a good performance in the field of machine learning such as image recognition, speech recognition, and natural language processing. The biggest difference between deep learning and machine learning lies in feature engineering. Feature engineering is to express expert knowledge in professional fields in specific features, to reduce the complexity of data, and to generate data patterns that algorithms can handle. In machine learning, most applications require manual feature engineering, which requires a large amount of expert knowledge to encode the original data into characteristic data formats, such as the length of the URL, and whether certain keywords are included in the URL. Deep learning does not require such artificial feature engineering. The model directly acquires deep features from the data. This is the biggest difference between deep learning and traditional machine learning methods. Therefore, we are concerned about whether we can use the appropriate deep learning model to automatically extract the structure and semantics features in the phishing website URL, and then use these features to distinguish phishing website URLs from benign website URLs.

3.2. Basic Idea of PDRCNN

3.2.1. Problem Definition. We treat the phishing URL as a string, and the phishing website detection problem is equivalent to the text categorization problem. In our proposed method, we follow the machine learning method to detect the phishing website, and regard the phishing website detection problem as a classification problem. With the deep learning method, in the training process of the model, the neural network can extract the intrinsic feature expression in the URL data, and then classify the website into phishing websites.

3.2.2. The Structure of PDRCNN. Figure 2 shows the structure of the PDRCNN method. The input of PDRCNN is a URL string, and the output is whether the URL belongs to a

Table 1: Comparison of PDRCNN with related works.

Type	Work	Search engine dependence	Third-party dependence	Language dependence	Number of experimental samples
Blacklist/ whitelist-based	Google Safe Browsing API [2]	No	Yes	No	–/–
	AIWL [3]	No	No	No	16/18
Visual similarity-based	Doom Tree similarity [4]	No	No	No	8/320
	BaitAlarm [5]	Yes	Yes	No	0/300
	LinkGuard [6]	No	Yes	No	0/8
	Phishdentity [7]	Yes	Yes	No	5000/5000
Heuristic-based	CANTINA [8]	Yes	Yes	Yes (English)	100/100
	PhishNet [9]	No	Yes	No	0/6000
	Finite state machine [10]	Yes	No	No	99/25
	New approach [11]	Yes	Yes	Yes (English)	1200/3374
	PhishShield [12]	No	Yes	No	250/1600
	PDA [13]	No	Yes	No	405/1120
Machine learning-based	Fuzzy logic [14]	Yes	Yes	No	0/606
	CANTINA+ [15]	Yes	Yes	Yes (English)	4883/8118
	Page classification [16]	Yes	No	No	200/325
	AC [17]	No	Yes	No	450/2500
	MCAC [18]	No	Yes	No	1350 (All)
	SMO [19]	No	Yes	Yes (Chinese)	1462/1416
	Phish detector [20]	Yes	Yes	No	1271/3066
	Know thy domain name [21]	No	Yes	No	2000/4013
	Metaheuristic algorithm [22]	Yes	Yes	No	8599/2456
	HEFS [23]	No	No	No	5000/5000
Deep learning-based	Classifying phishing URLs using RNN [24]	No	No	No	1000000/1000000
	Stacked autoencoder [26]	Yes	Yes	No	20000/17000
	Phishing detection with LSTM [25]	No	Yes	No	2000/2000

phishing website. After receiving a URL string, PDRCNN first encodes the URL as a string into the two-dimensional tensor of the fixed space and then passes the encoded tensor into the designed deep learning neural network. The model extracts the structural and semantic features in the URL, and then uses the Sigmod function to classify the extracted features and finally outputs the classification result of the URL.

3.3.3. Choice of Deep Learning Model. Typical deep learning models include CNN, RNN, autoencoders, and DBN (deep belief networks).

Among them, the RNN is good at processing sequence data, such as a consequent speech or a consequent text, and can well handle the problem of the connection between the data before and after the sequence. RNNs memorize the previous information and then apply it to the current calculation, that is, the nodes between the hidden layers are connected. And the input of the hidden layer includes the input, and the output of the layer includes the data of the hidden layer at the previous moment. Considering that we need to extract the structural and semantic features in the sequence of URL strings, we choose the bidirectional LSTM model in RNNs [31, 32].

For text information, in addition to semantics from front to back, semantic information is also included from the back to the front. The basic idea of bidirectional recurrent neural network (BRNN) is that each training sequence consists of two recurrent neural networks, and the result provides complete past and future context information for each point in the output layer sequence. The basic idea is that each training sequence has two cyclic neural networks: forward and backward. This result provides complete past and future context information for each point in the output layer sequence.

The CNN is another representative network structure in the deep learning method. It can extract the local features of the data well, and not only has great success in the field of image processing but also can deal with text classification problems. In 2014, Kim [33] proposed using CNN to deal with text classification. In 2015, Lai et al. proposed TextRCNN [34] to deal with the problem of text categorization and achieved very good classification results. The method proposed by them combines the RNNs and CNN. They use RNNs to replace the convolutional layer in the CNN model; that is, they use RNNS to extract the word representation of each character in the sequence, and then use the pooling layer to extract the entire text representation, and finally, it is classified by the classifier.

In the PDRCNN method, we combine RNNs and CNN to extract the intrinsic features in the URL string. Firstly, the recurrent structure in the PDRCNN method fills in the global features of the URL string to each of the characters, and the tensor passed into the convolutional structure no longer contains the original URL data. Then, we get the characteristics of the entire URL string through the convolutional layer and the pooling layer through three types of convolution kernels of different sizes.

Table 2: Nine artificial phishing website character features.

Features	Description
F1 [15]	*Embedded domain*: some phishing URLs will insert the benign website domain name into the domain name to hide the real domain name. For example, the following phishing link nests the domain name of eBay.com to confuse the user. http://cgi.ebay.com.ebaymotors.732issapidll.private99dll.qqmotorsqq.ebmdata.com
F2 [15]	*IP address*: this feature checks if a page â€™s domain is an IP address.
F3 [15]	*Number of dots in the URL*: this feature counts the number of dots in the URL. Phishing pages tend to use more than 5 dots in their URLs than the legitimate sites.
F4 [15]	*Suspicious URL*: the phishing link will confuse the user by inserting a special character in the URL. The commonly used special characters include "@," "-," etc., and "@" hides the phishing URL by commenting out the domain name that appears before its position. Benign links will not perform similar operations.
F5 [15]	*Number of sensitive words in the URL*: phishing websites add sensitive words to pretend to be legitimate websites. Sensitive words like "login" and "registered" can increase the similarity of phishing sites, allowing users to submit forms with private information.
F6 [15]	*Out-of-position top-level domain (TLD)*: some phishing websites often have strange top-level domains in their domain names. This is because the links contain "edu, cn, com" etc, which makes it easier to obtain the trust of users. Phishing attackers insert common top-level domains in domain names or paths. For example, http://www.inc-paypal-id.com.apps-web.cf/ uses a separator to insert a com in the domain name.
F7 [28]	*Length of the URL*: the URL of a phishing website is different in length from that of a legitimate websites. We set the threshold to 54, and the URL length is greater than 54. It is more likely to be a phishing website.
F8 [29]	*Number of "/" s in the URL*: this feature counts the number of dots in the URL. Phishing pages tend to use more than 5 "/" in their URLs than the legitimate sites.
F9 [30]	*Number of sensitive domain in the URL*: paypal.com, apple.com, google.com, eBay.com, eBay.it, maybank2u.com, aol.com, yahoo.com, nab.com, natwest.com, amazon.com, bt.com, Alibaba.com, facebook.com, key.com

4. PDRCNN Design

4.1. Data Preprocessing. Data preprocessing is based on word embedding, which encodes the URL string into a two-dimensional tensor that can be received by the deep learning model. After data preprocessing, each character is encoded to a fixed length vector consisting of 0 and 1. This is because the neural network needs to ensure that the input data is a vector of numbers when performing mathematical operations.

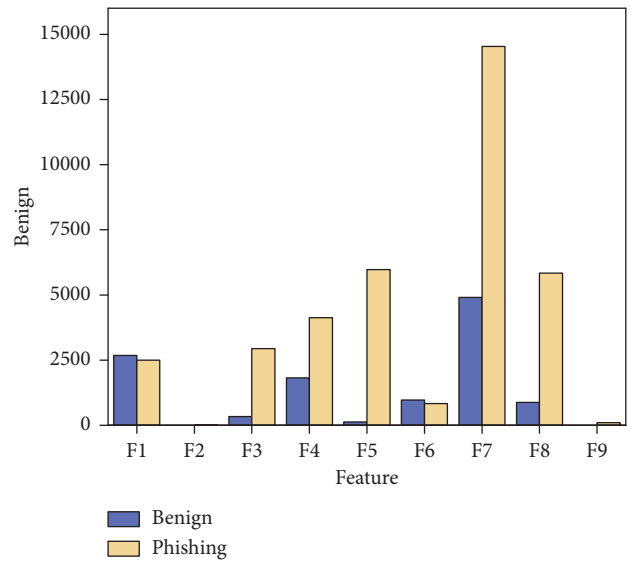

Figure 1: 9 URL feature statistics.

First, we process the length of the URL string. There is a limit on the length of the URL in the HTTP standard protocol RFC2616 document: "Servers ought to be cautious about depending on URL lengths above 255 bytes because some older client or proxy implementations might not properly support these lengths." So, we set the length of URL to 255 characters, which means that if the length of the URL exceeds 255 characters, only the first 255 characters are intercepted. If the length of the URL is shorter than 255, add 0 to the end of the URL string to a length of 255 characters.

At the same time, we counted the frequency of occurrences of characters in all URLs in the dataset and selected the first 59 characters with the highest frequency as valid characters. It contains 26 English letters, 10 Arabic numerals, and 23 special characters including "@/: = #-." Other characters that are not in the list are all "special characters," and each URL is treated as a sequence of only 60 different characters. As shown in Figure 3, each character is encoded into a 60-bit 01 string where one in the interface value row and zero in the rest. Then, we use the word2vec method in natural language processing to encode the previously processed 60-bit 01 string into a 64-bit word vector. Thus, each URL is processed into a two-dimensional matrix of length $255 * 64$, which then passes to the input of PDRCNN.

4.2. Recurrent Convolutional Neural Network. As shown in Figure 4, in PDRCNN, we combine the RNN and the CNN to extract the intrinsic structural and semantic features of the preprocessed URL.

The input of the deep learning model in the PDRCNN method is a two-dimensional tensor $X = \{x_1, x_2, x_3, \ldots, x_{255}\}$, where x_i is a vector consisting of 64 zeros or ones. $y_r(x)$ is the output of the recurrent structure and is also the input to the convolutional structure. $y_c(x)$ is the output of the convolutional structure.

Recurrent structure extracts the features in the URL by bidirectional LSTM neural network, including forward pass and backward pass. $y_r^f(x)$ and $y_r^b(x)$ are obtained after X

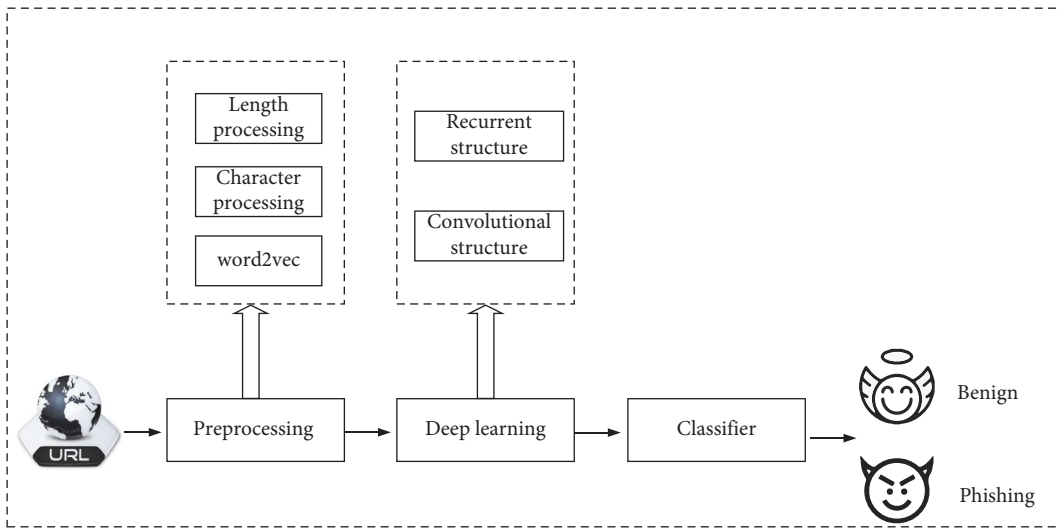

Figure 2: Workflow of PDRCNN.

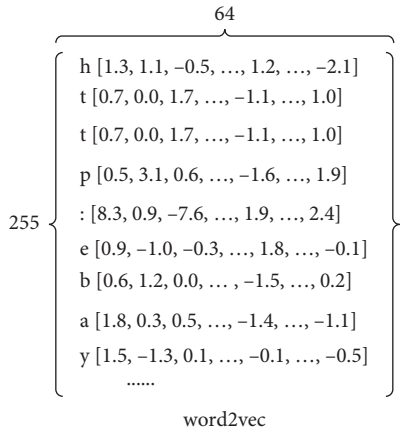

Figure 3: Data preprocessing.

treatment. The recurrent structure output is $y_r(x) = \text{Concat}(y_r^f(x), y_r^b(x))$, which is a $255 * 128$ tensor. Among them,

$$y_r^f(x_i) = \sigma_f \cdot \tanh\left(W_{rnn}^{(f)} y_r^f(x_{i-1}) + W_{rnn}^{(sf)} x_i\right), \quad (1)$$

$$y_r^b(x_i) = \sigma_b \cdot \tanh\left(W_{rnn}^{(b)} y_r^b(x_{i+1}) + W_{rnn}^{(sb)} x_i\right). \quad (2)$$

The calculation process of the character $y_r^f(x)$ from front to back is as shown in equation (1), where $W_{rnn}^{(f)}$ represents the parameter matrix in the neural network, corresponding to the forget gate in the LSTM model, and the tensor in the network is transferred from the hidden layer to the next hidden layer. $W_{rnn}^{(sf)}$ is used to combine the semantic features of the current character into the feature vector, corresponding to the input gate in the LSTM. x_i and $y_r^f(x_{i-1})$, respectively, represent the semantic features of the current character and the previous character. The first character of all URLs only contains its own feature information. tanh is a nonlinear activation function that provides RNN with the ability to handle nonlinear problems. σ_f represents the output of each cell when passing features from front to back. The feature $y_r^b(x)$ from the back to the front is similar to the feature calculation process from the front to the back, as in equation (2), where the last character of all URLs contains only its own feature information.

Convolutional structure can be divided into two stages: In the first step, the local features in the tensor are extracted by the multilayer convolution layer. Here, we select three types of convolution kernels W_{cnn}^k of different sizes, each of which contains 32 convolution kernels of the same size. The sizes of these three types of convolution kernels are $5 * 120$, $6 * 120$, and $7 * 120$. The second step uses maxi-pooling to activate the features generated by the convolutional layer, extracts the most representative features of the URL, and splices the results of the convolution and pooling of the three types of convolution kernels together to form the final feature vector $y_c^k(x)$, as in equation (3). Finally, the results of the three-layer convolution and pooling layer processing are connected as a one-dimensional tensor $y_c(x)$, as in equation (4).

$$y_c(x) = \text{Concat}\left(y_c^k(x)\right) k \in \{5, 6, 7\}, \quad (3)$$

$$y_c^k(x) = \text{MaxPooling}\left(\text{relu}\left(\text{Conv}\left(W_{cnn}^k, y_r(x)\right) + b_{cnn}\right)\right). \quad (4)$$

4.3. Classifier. Once we extract the feature vector $y_c(x)$ in the URL, we use the fully connection layer and the sigmoid function to distinguish the URL into the benign and the phishing website, as in equation (5). Logit indicates the probability that the URL calculated by the PDRCNN method belongs to the phishing website. We set 0.5 to determine the threshold of positive and negative samples. The output probability is less than 0.5, indicating that the URL belongs to the benign website. If the output probability is greater than or equal to 0.5, the URL belongs to the phishing website.

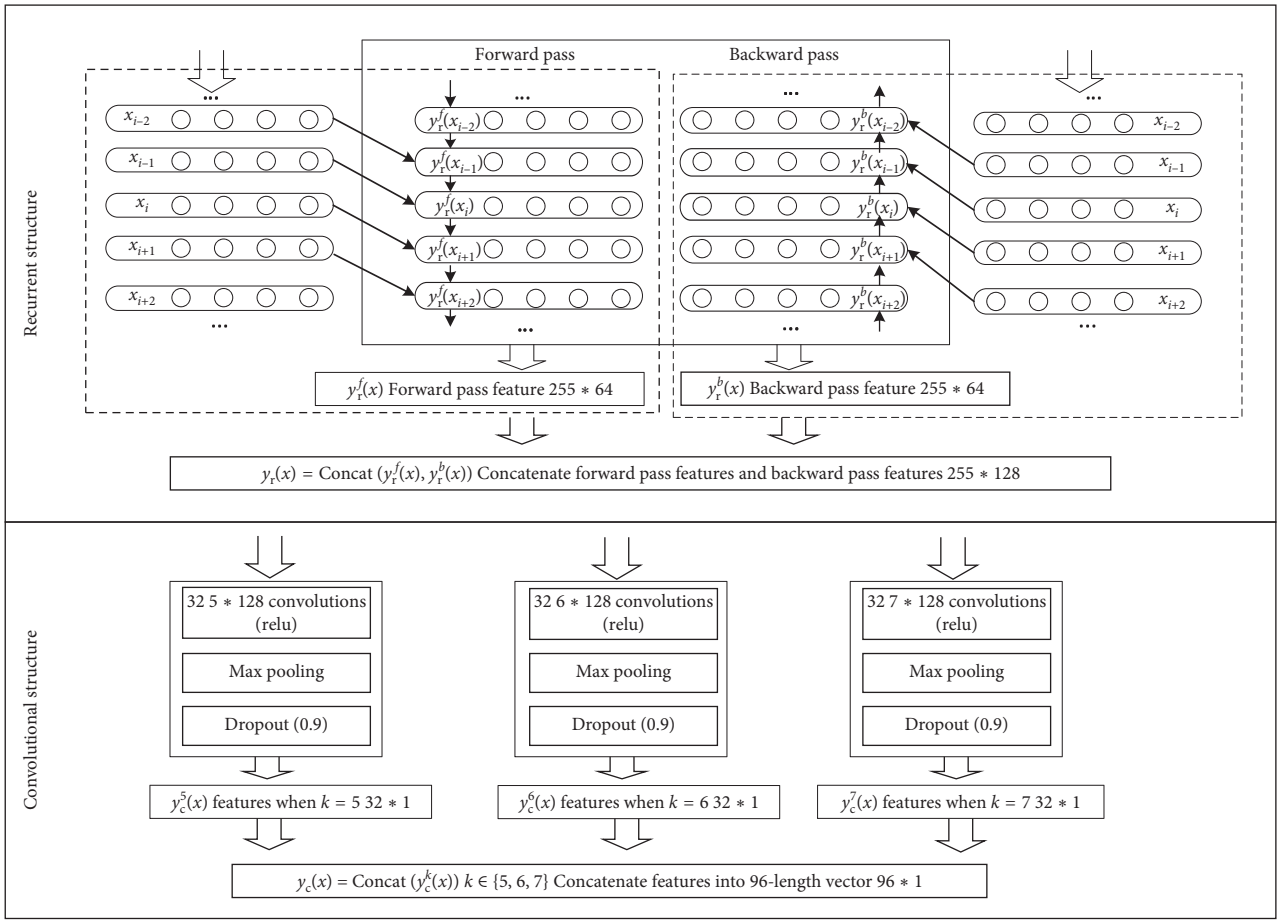

Figure 4: PDRCNN deep learning model structure.

$$\text{logit} = W_{\text{logit}} \cdot y_c(x) + b_{\text{logit}}. \tag{5}$$

4.4. Training. We define all of the parameters to be trained as $\theta = \left\{W_{\text{rnn}}^{(f)}, W_{\text{rnn}}^{(sf)}, W_{\text{rnn}}^{(b)}, W_{\text{rnn}}^{(sb)}, W_{\text{cnn}}^{k}, b_{\text{cnn}}, W_{\text{logit}}, b_{\text{logit}}\right\}$ we chose cross entropy as the loss function and trained the model parameters by minimizing the cross entropy. First, use the nonlinearization approach sigmoid to study logit, as in equation (5), and then calculate the loss between the PDRCNN output and the actual label, as in the following equation:

$$p_i = \text{sigmoid}\left(\text{logit}_i\right) = \frac{1}{1 + e^{-\text{logit}_i}},$$

$$\text{loss}\left(y_{\text{label}}, p_i\right) = -\frac{1}{N}\sum_{i}^{N} \cdot \left[y_{\text{label}} \cdot \ln p_i + \left(1 - y_{\text{label}}\right) \cdot \ln\left(1 - p_i\right)\right]. \tag{6}$$

Finally, the Adam (adaptive moment estimation) optimizer is chosen to minimize the loss and make the model converge. The Adam algorithm dynamically adjusts the learning rate for each parameter based on the first-order moment estimate and the second-order moment estimate of the gradient of each parameter based on the loss function. We chose Adam because the learning step size of each iteration parameter has a certain range, and will not cause a large learning step because of a large gradient and the parameter value is relatively stable. We set the learning rate to 0.01. After each optimizer performs gradient descent optimization, the parameters in PDRCNN are updated. When the loss value is reduced enough, the model converges and the training ends.

5. Experiment

5.1. Dataset. We obtained the URL data of all phishing websites published from August 2006 to March 2018 from the PhishTank website, with a total of 5,118,727 URL data. We use the crawler program to determine whether these URLs are valid, remove URLs that are not surviving or have errors in the content of the web page and finally get 245,385 valid phishing URLs.

For the data of the benign website URL, we first obtain the top 1 million domain of the Alexa website domain name ranking. Since these domains are normal website homepage domains, in order to be more general, we use search engines to search for these domain names and obtain the URLs of the top 10 links for each search, retain the surviving links, and perform de-reprocessing to get 245023 benign URLs.

There are two points to note about the processing of data:

(1) In order to improve the quality of the benign URL data, we use the search engine to make the data more generalized, instead of directly using the homepage URL of the top-ranked domain on Alexa as the benign website data set. The homepage URL corresponding to the domain name is relatively short in length and generally has only one level directory. In contrast, phishing website URLs are basically multilevel directories in structure and are relatively long in length. If the URL of the homepage corresponding to the top-ranked domain of Alexa is directly used as the benign website data set, the phishing website and the benign website can be accurately distinguished in the number of directories and the length of the URL.

(2) In the comparison experiment, the CANTINA+ method needs to rely on the content of the web page. In order to ensure the consistency of the experimental data, we use the crawler to crawl the website corresponding to the collected URL and remove the URL and web page that are not surviving or have errors in the html content.

We divide the data set into a training set, a validation set, and a testing set in a ratio of 8 : 1 : 1, that is, we use 4/5 of the data as the training set to train the hyperparameters of the PDRCNN model, including the weightsand biases of each unit. Offset, 1/8 of the data is used as a validation set to adjust hyperparameters in the neural network, such as the number of hidden layers and unit size of the RNN, and the rest of the data is used as a testing set to predict the classification results. The sample size of each set is detailed in Table 3.

5.2. Evaluation Indicators. We use the Python 3.6 and tensorflow to implement the PDRCNN, and use the third-party module scikit-learn in python to calculate the following eight data indicators to evaluate the advantages and disadvantages of PDRCNN and other methods: accuracy, precision, recall, *F*-measure, ROC curve, AUC value, training time, and test time.

5.2.1. Accuracy. Accuracy is the ratio of the total number of correctly classified samples in the test set to the total number of samples. In our experiments, it refers to the ratio of the benign website URL being correctly judged to be benign and the phishing website URL being correctly judged as the total number of phishing and the total number of test sets.

5.2.2. Precision. The ratio of the number of phishing website URLs correctly judged by the model to the number of phishing website URLs.

5.2.3. Recall. The ratio of the URL of the test phishing website is correctly judged as the phishing website accounting for the URL of all phishing websites.

Table 3: Dataset statistics.

Label	Name	Training set	Validation set	Testing set
1	Phishing	196308	24538	24539
0	Benign	196019	24502	24502
All		392327	49040	49041

5.2.4. F-Measure. There are sometimes contradictions in the precision rate and the recall rate, and it is necessary to consider them comprehensively. The *F*- measure is a weighted harmonic average of the precision rate and the recall rate. The higher the *F*- measure, the more effective the method.

These metrics are calculated as follows:

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{F} &= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \end{aligned} \tag{7}$$

Among them, TN indicates that the benign website URL is correctly marked as a benign website, TP indicates that the phishing website URL is correctly marked as a phishing website, FN indicates that the phishing website URL is incorrectly marked as a benign website, and FP indicates that the benign website URL is incorrectly marked as a phishing website.

The ROC (receiver operating characteristic) curve and AUC are often used to evaluate the merits of a binary classifier. The horizontal coordinate of the ROC curve is FPR, indicating the probability that the normal website URL is incorrectly tagged as a phishing website; the ordinate is TPR, which indicates the probability that the phishing website URL is correctly labeled as a phishing website. Their definitions are as follows:

$$\begin{aligned} \text{FPR} &= \frac{\text{FP}}{\text{FP} + \text{TN}}, \\ \text{TPR} &= \frac{\text{TP}}{\text{FP} + \text{TN}}. \end{aligned} \tag{8}$$

It can be known from the formula that the closer the ROC curve is to the upper left corner, the better the performance of the classifier. The AUC value refers to the area under the ROC curve, and the AUC value ranges between 0.5 and 1. As an image, the ROC curve does not very clearly indicate which classifier is better in many cases, and as a numerical value, a larger AUC value can directly indicate that the classifier is better.

Training time is the time required for PDRCNN to extract features and determine optimal neuron parameters on the training set. For machine learning methods, the training time includes the time for feature extraction of training set samples and training of machine learning algorithms.

The test time refers to the time required for the classification result to be detected for each sample on the test set after the PDRCNN training is completed. For the machine learning method, the test time also includes the time of feature extraction of the test set sample and the classification of the machine learning algorithm.

5.3. PDRCNN Parameters Optimization. In the neural network structure, the setting of some hyperparameter values is crucial. The number of hidden layer in RNN and the convolution kernel size of CNN play an important role in the classification accuracy. The number of hidden layer was chosen from the set {8, 16, 32, 64, 128}, and we set the size of the convolution kernel in the range of 2 to 7, and then sort and combine the convolution kernel according to the accuracy and loss. The size of epoch and batch are also important if epoch is too small, as PDRCNN cannot achieve the highest accuracy and there may be overfitting. We set the epoch from 1 to 40, and choose the batch size from the set {64, 128, 256, 512, 1024, 2048, 4096}. After hyperparameters training in the training phase and verification set adjustment, the optimal hyperparameters of the PDRCNN method are follows: the number of hidden layer units in RNN is 64, the convolution kernel size of CNN is {5, 6, 7}, and the epoch size is 32 and batch size is 2048.

First, we tested the effects of different number of hidden layers in RNN on the validation set. The loss and accuracy are shown in Table 4. It can be seen from the table that when the number of units increases from 8 to the next, the correct rate is continuously increased, but after more than 64, the correct rate is reduced, and the loss is increased.

Next, by fixing the number of hidden layers to 64, we tested the influence of the size of the convolution kernel. We first use a single convolution kernel and sort the effects of convolution kernels of different sizes, and then combine them in turn. As shown in Table 5, when a single convolution kernel is used, the classification effect of the verification set is sorted from high to low, and the convolution kernel size sorting result is: 6, 5, 7, 4, 3, and 2. After combining, it can be found that the best results are obtained when the convolution kernel size is {5, 6, 7}.

Then, we compare the effect of different batch sizes on the correct rate and loss of the model. As shown in Table 6, when the batch size is set to 2048, the model has the highest accuracy and the least loss.

Finally, we set the number of hidden layers in RNN to 64, the convolution kernel size to {5, 6, 7}, and the batch size to 2048, comparing the effects of different epoch sizes on the accuracy of the method. As shown in Figure 5, when the epoch is 32, the model obtains the minimum loss, and when the epoch is increased, the loss does not decrease, and it is in a stable equilibrium state.

5.4. Baseline Models. To verify PDRCNN's ability to identify phishing websites, we implemented four baseline models for comparison:

Table 4: Effect of the number of hidden layers in RNN.

Hidden layers	Accuracy (%)	Loss
8	93.48	2.25
16	94.53	1.89
32	95.0	1.73
64	**95.61**	**1.52**
128	95.12	1.68

Table 5: Effect of convolution kernel size in CNN.

The convolution kernel	Accuracy (%)	Loss
2	94.56	1.88
3	94.68	1.84
4	94.87	1.80
5	95.09	1.69
6	95.12	1.68
7	95.03	1.72
5, 6	94.99	1.73
5, 6, 7	**95.61**	**1.52**
4, 5, 6, 7	95.13	1.68
3, 4, 5, 6, 7	95.2	1.66
2, 3, 4, 5, 6, 7	95.27	1.64

Table 6: Effect of the number of batch size.

Batch size	Accuracy (%)	Loss
64	92.41	2.62
128	93.72	2.17
256	94.52	1.89
1024	95.28	1.63
2048	**95.61**	**1.52**
4096	95.15	1.67

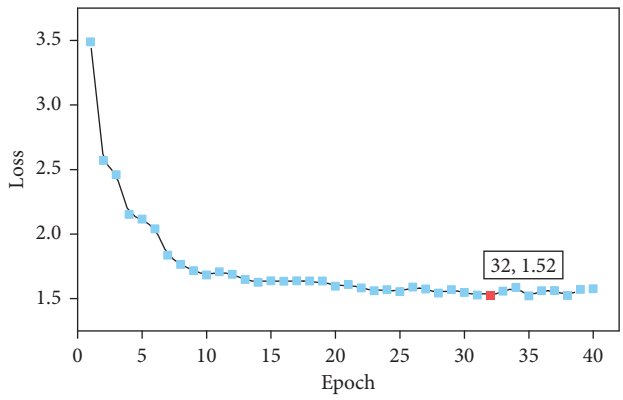

Figure 5: Effect of the number of epoch size.

(1) Replace the deep learning model in the PDRCNN method with a separate RNN and CNN, where the hyperparameter value of the model is the same as PDRCNN.

(2) CANTINA+ [14], is a machine learning method proposed by Stanford for identifying phishing websites. They have proposed 15 features, including 6 URL character-level features, 4 html page features

and 5 other features provided by third-party organizations and search engines.

(3) Standard n-gram feature vector extraction method: In the embedding process of the PDRCNN method, we encode the URL into a string consisting of 60 different characters. We have chosen the 2-bit BiGram method (two sized n-grams).

(4) Finally, based on the character-level features of the nine URLs proposed by researchers in the existing research, these features include statistical knowledge and whether sensitive words appear in the URL.

After extracting the BiGram method and 9 URL character-level features, the test set is performed using three standard machine learning classification algorithms, the Naive Bayesian algorithm (GaussianNB), the Logistic Regression Algorithm (LG), and the Gradient Boosting Decision Tree (GBDT).

5.5. Experiment Results. In order to evaluate the performance of the PDRCNN, we used a 10-fold cross-validation strategy. This process consists of splitting data in 10 folds. Then, train the data using two folds while the remaining one is used for model validation. This process is repeated 10 times, only using each fold for validation once. Table 7 shows the results of the 10-fold cross-validation.

We used the established training set and test set to test the comparison of the PDRCNN method with the four baseline models. Table 8 lists the test results of PDRCNN on the test set. According to the confusion matrix, we can find that in all the phishing website URL test sets, there are 23,013 phishing website URLs correctly classified as phishing, and only 632 normal website URLs are incorrectly judged as phishing website URLs, and FPR is only 2.6%.

Using the statistics in Figure 1, we analyzed the reasons why the 632 benign websites were misjudged as phishing websites. When a sensitive word such as "login" or "registered" appears in the URL, our detection engine is more likely to prefer the benign website URL to the phishing website URL. In the benign website URL test data set, there are 126 URLs containing these sensitive words, of which 19.8% of the URLs are misjudged as phishing websites, and only 2.5% of the URLs that do not contain these sensitive words are misjudged as a phishing website. The same is true for the other eight features mentioned in 3.1. When the URL feature is 0, about 2.5% of the data is misjudged as a phishing website.

We also analyzed the reasons why 1,525 phishing websites were missed as benign websites. That is, when the URL is short, the detection engine is more likely to miss the judgment. As shown in Figure 1, the number of benign website F7 features is only 48,532, while the statistic data in the phishing website data set is 145,384. The URL of the benign website is indeed shorter, and the phishing website URL may need to be the URL containing the brand name of the benign website that you want to model, such as "apple," "microsoft," and "google," so the length of the URL will be longer, which is also a limitation of the method of detecting the phishing website by the URL.

Figure 6 shows the comparison result between the PDRCNN method and the simple RNN and CNN, Figure 7 shows the comparison result between the PDRCNN model and the CANTINA+ method, and Figure 8 shows the comparison result between the PDRCNN model and the BiGram and 9-bit URL character-level features result. From the ROC curve, it can be found that PDRCNN is closer to the upper left of the coordinate axis than the other four baseline models, which means that it can have a higher true positive rate while ensuring a lower false positive rate. This shows that the dominant performance of PDRCNN is more obvious on the AUC value. The AUC value of the PDRCNN model is as high as 99%, followed by the RNN and CNN models. This shows that the PDRCNN model combined with RNN and CNN can effectively combine the advantages of the two deep learning models. On the other hand, it also shows that the deep learning model can make good use of the URL string of the website to detect phishing websites. This is followed by the BiGram method and the CANTINA+ method. After the BiGram method extracts the feature vector, different machine learning methods have different performance, which indicates that the naive Bayesian algorithm (GaussianNB) and the gradient lifting decision tree algorithm (GBDT) are compared to the logic. The regression algorithm (LG) is able to better learn the features in the vector.

Finally, we calculate the performance of PDRCNN and the four baseline models in terms of accuracy, precision, recall, F-measure, AUC value, training time, and test time, as shown in Table 9. In training time, the PDRCNN model takes longer than the separate RNN and CNN, 9-bit URL character-level feature methods. This is because our method needs to train more parameters, and CANTINA+ relies on the results of third-party organizations and search engines, so it consumes a lot of time in feature extraction, so it takes a long training time and testing time. The feature vector of each URL extracted by the BiGram method has 3600 dimensions. Because the dimension is too large, it takes a lot of time to use the machine learning method for training. In terms of test time, PDRCNN has obvious advantages over the other four baseline models. This is because our method does not rely on the results of third-party organizations and search engines, and the feature dimensions extracted by the model are compressed into 96-dimensional, so the test time is short.

5.6. The Effect of 9 URL Features. In the experiment, we considered whether to incorporate the 9 URL character-level features into the deep learning model to help improve the accuracy of PDRCNN in detecting the phishing website URL, so we did the corresponding experiment.

First, after receiving the URL data, we extract the 9 character features of the data, enter a fully connected layer, and expand the 9 features into a 32-bit vector. The 36-bit vector is then concatenated to the 96-bit vector extracted from the neural network in PDRCNN, and then input to the

Table 7: Results of the 10-fold cross-validation.

Fold	Accuracy (%)	Precision (%)	Recall (%)	*F*-measure (%)	AUC (%)
1	95.86	97.15	94.49	95.8	99.09
2	95.93	97.41	94.38	95.87	99.04
3	95.69	96.53	94.78	95.65	98.99
4	96.01	97.26	94.7	95.96	99.07
5	95.62	97.36	93.78	95.54	98.99
6	95.81	97.03	94.52	95.76	98.99
7	95.86	97.12	94.54	95.81	99.06
8	95.64	97.48	93.7	95.55	98.98
9	95.83	98.07	93.51	95.74	99.1
10	95.69	97.29	94.01	95.62	98.97
Average	**95.79**	**97.27**	**94.24**	**95.73**	**99.03**

Table 8: Confusion matrix of experimental results.

	Predicted as phishing	Predicted as benign	Total
Phishing	23,014	1,525	24,539
Benign	632	23,870	24,502
Total	23,646	25,395	49,041

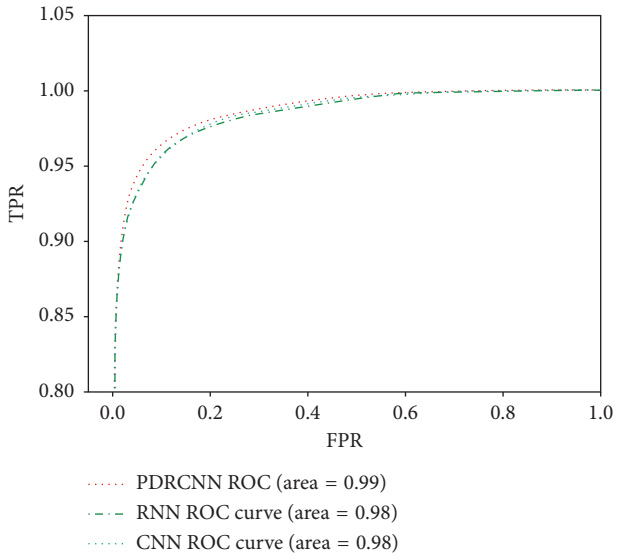

Figure 6: PDRCNN compared with RNN and CNN.

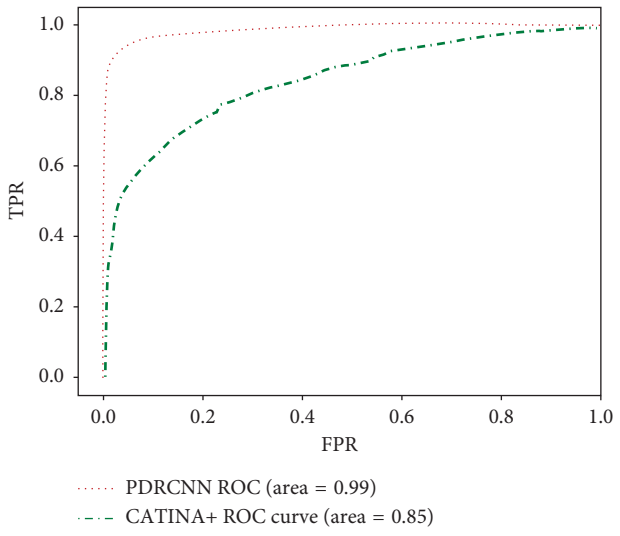

Figure 7: PDRCNN compared with CANTINA+.

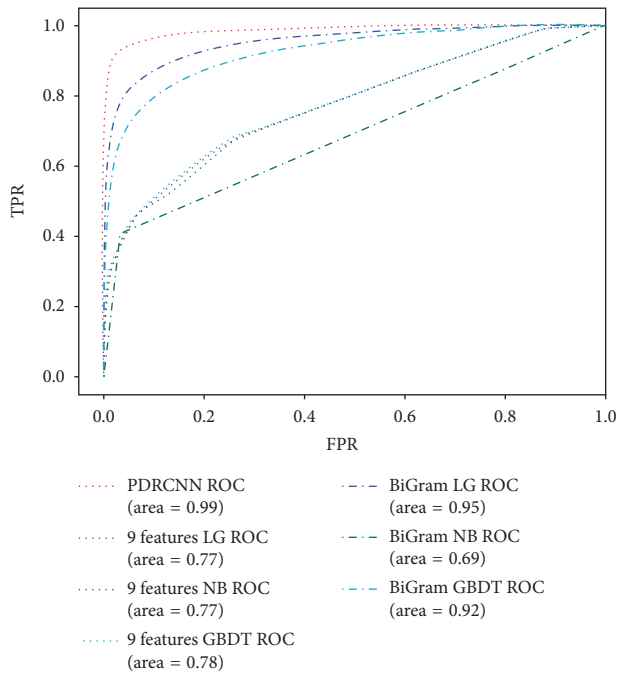

Figure 8: PDRCNN compared with BiGram and 9 features.

Table 9: Performance comparison with 4 baseline.

Method	Accuracy (%)	Precision (%)	Recall (%)	*F*-measure (%)	AUC (%)	Training time (s)	Test time (s)
PDRCNN	**95.6**	**97.33**	**93.78**	**95.52**	**98.96**	**4426.15**	**40.66**
RNN	94.24	95.14	93.26	94.19	98.33	2033.92	17.85
CNN	94.46	95.52	93.31	94.4	98.48	442.79	4.47
CANTINA+	76.43	78.24	76.98	76.44	85.43	133h	30h
9 features with GaussianNB	64.91	94.24	31.81	47.57	77.93	108.5	20.64
9 features with LG	71.2	75.37	63.05	68.66	78.21	107.1	20.81
9 features with GBDT	71.51	75.38	63.94	69.19	78.51	148.21	21.13
BiGram with GaussianNB	68.69	92.3	40.82	56.61	69.71	2985.94	638.52
BiGram with LG	88.68	90.96	85.91	88.36	95.68	2982.34	652.74
BiGram with GBDT	84.55	88.32	79.65	83.76	92.55	26420.99	639.12

Then, we use the ROC curve and the AUC value to evaluate the PDRCNN model and the four baseline models.

Table 10: Performance comparison with 9 URL features added.

Method	Accuracy (%)	Precision (%)	Recall (%)	*F*-measure (%)	AUC (%)	Training time (s)	Test time (s)
PDRCNN	95.6	97.33	93.78	95.52	98.96	4426.15	40.66
PDRCNN and 9 artificial features	95.48	97.39	93.46	95.39	98.92	4415.23	40.97

final classifier. As shown in Table 10, even if the 9 URL character-level features is added to the model in PDRCNN, the *F*-value and AUC value of the model on the test set are not improved. This explains to a certain extent that the 96-bit feature extracted by PDRCNN already contains the 9-bit URL character-level feature, so even increasing the character-level features proposed by the researchers does not help improve the accuracy of the deep learning model.

5.7. Robustness. In addition to the comparison of the PDRCNN method with the four baseline models in the evaluation indicators, we also tested the robustness of the PDRCNN method. First, the phishing website URL is divided according to the publication time on the PhishTank website, and the benign website URL is randomly divided according to the amount of data published by the phishing website every year. Then, use the URL published a year ago as the training set, and the URL published in the year is tested as a test set. For example, use the phishing website URL published before 2014 and the same number of benign website URLs as the training set, that is, a total of 72,232 effective phishing website URLs published by PhishTank in 2006, 2007, 2008, 2009, 2010, 2011, 2012, and 2013. The website URL and 70,000 benign website URLs are used as training sets. In 2014, a total of 24,501 phishing website URLs and 24,000 benign websites were published as test sets. PhishTank has published phishing website data since 2006,

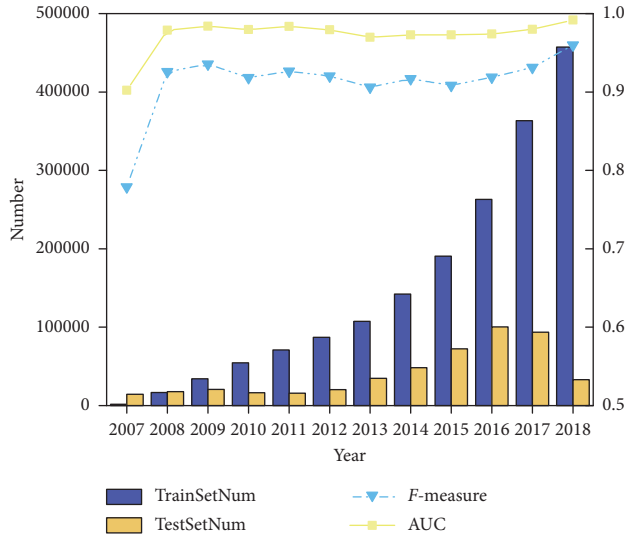

Figure 9: Result of robust experiment.

so our robustness test includes 12 test results from 2007 to 2018. As shown in Figure 9, with the increase of the amount of data in the training concentration every year, the F value and AUC value of PDRCNN show an increasing trend year by year, which shows that our method is robust.

6. Conclusion

To the best of our knowledge, we are the first one who use the deep learning model to detect phishing in the context of cybersecurity issues, and the first who use hundreds of thousands of phishing URLs and normal website URLs for training and testing. The experimental results showed that compared with the existing research, PDRCNN can detect the URL of the phishing website without relying on third-party data and search engines, with a highest classification accuracy among other models.

In our experiments, the main problem was that the training time was too long, but the trained PDRCNN model was far ahead of the existing research in terms of test time and accuracy. There are some other potential drawbacks to the classifier. One obvious disadvantage is that when the phishing website URL itself does not have relevant semantics, PDRCNN will not be able to classify correctly, and PDRCNN does not care whether the website corresponding to the URL is alive and if there is an error. Therefore, when applying PDRCNN to the actual detection scenario, it is necessary to verify the validity of the URL in advance.

Data Availability

The experiment data reported in the paper can be acquired from the corresponding author through emails.

Conflicts of Interest

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

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant nos. 61672543 and 61772559 and the Open Research Fund of Hunan Provincial Key Laboratory of Network Investigational Technology under Grant nos. 2017WLZC002 and 2017WLZC003.

References

[1] APWG, "Anti phishing work group," 2019, https://www.antiphishing.org/.

[2] Google, "Google safe browsing API," 2019, https://developers.google.com/safe-browsing/v4/.

[3] 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, Alexandria, VA, USA, October 2008.

[4] W. Liu, G. Huang, X. Liu, M. Zhang, and X. Deng, "Detection of phishing webpages based on visual similarity," in *Proceedings of the Special Interest Tracks and Posters of the 14th International Cconference on World Wide Web*, pp. 1060-1061, ACM, Chiba, Japan, May 2005.

[5] J. Mao, L. Pei, K. Li, W. Tao, and Z. Liang, "Baitalarm: detecting phishing sites using similarity in fundamental visual features," in *Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 790–795, IEEE, Xi'an, China, September 2013.

[6] 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.

[7] K. L. Chiew, J. S.-F. Choo, S. N. Sze, and K. S. C. Yong, "Leverage website favicon to detect phishing websites," *Security and Communication Networks*, vol. 2018, Article ID 7251750, 11 pages, 2018.

[8] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pp. 639–648, ACM, Banff, Canada, May 2007.

[9] P. Prakash, M. Kumar, R. Rao Kompella, and M. Gupta, "Phishnet: predictive blacklisting to setect phishing attacks," in *Proceedings of 29th IEEE Conference on Computer Communications (Infocom)*, pp. 1–5, Citeseer, San Diego, CA, USA, March 2010.

[10] H. Shahriar and M. Zulkernine, "Trustworthiness testing of phishing websites: a behavior model-based approach," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1258–1271, 2012.

[11] G. Ramesh, I. Krishnamurthi, and K. S. S. Kumar, "An efficacious method for detecting phishing webpages through target domain identification," *Decision Support Systems*, vol. 61, pp. 12–22, 2014.

[12] R. S. Rao and S. T. Ali, "Phishshield: a desktop application to detect phishing webpages through heuristic approach," *Procedia Computer Science*, vol. 54, pp. 147–156, 2015.

[13] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 9, 2016.

[14] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7913–7921, 2010.

[15] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: a feature-rich machine learning framework for detecting phishing websites," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 2, pp. 1–28, 2011.

[16] M. He, S.-J. Horng, P. Fan et al., "An efficient phishing webpage detector," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12018–12027, 2011.

[17] R. M. Mohammad, L. McCluskey, and F. Thabtah, "Intelligent rule-based phishing websites classification," *IET Information Security*, vol. 8, no. 3, pp. 153–160, 2014.

[18] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948–5959, 2014.

[19] D. Zhang, Z. Yan, H. Jiang, and T. Kim, "A domain-feature enhanced classification model for the detection of Chinese phishing e-Business websites," *Information & Management*, vol. 51, no. 7, pp. 845–853, 2014.

[20] M. Moghimi and A. Y. Varjani, "New rule-based phishing detection method," *Expert Systems with Applications*, vol. 53, pp. 231–242, 2016.

[21] H. Shirazi, *Unbiased phishing detection using domain name based features*, Ph.D. thesis, Colorado State University, Fort Collins, CO, USA, 2018.

[22] M. Babagoli, M. P. Aghababa, and V. Solouk, "Heuristic nonlinear regression strategy for detecting phishing websites," *Soft Computing*, vol. 23, no. 12, pp. 4315–4327, 2018.

[23] K. L. Chiew, C. L. Tan, K. S. Wong, S. C. Y. Kelvin, and K. T. Wei, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," *Information Sciences*, vol. 484, pp. 153–166, 2019.

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

[25] W. Chen, W. Zhang, and Y. Su, "Phishing detection research based on LSTM recurrent neural network," in *Proceedings of International Conference of Pioneering Computer Scientists, Engineers and Educators*, pp. 638–645, Springer, Zhengzhou, China, September 2018.

[26] M. Nivaashini and R. S. Soundariya, "Deep stacked autoencoder based feature repsentation for phishing URLs detection," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 9, no. 6, pp. 904–916, 2017.

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

[28] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," 2010, https://arxiv.org/pdf/1009.2275.pdf.

[29] V. S. Lakshmi and M. S. Vijaya, "Efficient prediction of phishing websites using supervised learning algorithms," *Procedia Engineering*, vol. 30, pp. 798–805, 2012.

[30] H. Huang, L. Qian, and Y. Wang, "A SVM-based technique to detect phishing URLs," *Information Technology Journal*, vol. 11, no. 7, pp. 921–925, 2012.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, https://arxiv.org/abs/1605.05101.

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

[34] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, vol. 333, pp. 2267–2273, Austin, TX, USA, January 2015.