IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received December 31, 2018, accepted January 16, 2019, date of publication January 29, 2019, date of current version March 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2895751

Detecting Malicious URLs via a Keyword-Based Convolutional Gated-Recurrent-Unit Neural Network

WENCHUAN YANG, WEN ZUO, AND BAOJIANG CUI
School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: Wen Zuo (shiyu.nm@163.com)

This work was supported by the National Natural Science Foundation of China under Grant U1536122 and Grant 61502536.

ABSTRACT With the continuous development of Web attacks, many web applications have been suffering from various forms of security threats and network attacks. The security detection of URLs has always been the focus of Web security. Many web application resources can be accessed by simply entering an URL or clicking a link in the browser. An attacker can construct various web attacks such as SQL, XSS, and information disclosure by embedding executable code or injecting malicious code into the URL. Therefore, it is necessary to improve the reliability and security of web applications by accurately detecting malicious URLs. This paper designs a convolutional gated-recurrent-unit (GRU) neural network for the detection of malicious URLs detection based on characters as text classification features. Considering that malicious keywords are unique to URLs, a feature representation method of URLs based on malicious keywords is proposed, and a GRU is used in place of the original pooling layer to perform feature acquisition on the time dimension, resulting in high-accuracy multicategory results. The experimental results show that our proposed neural network detection model is very suitable for high-precision classification tasks. Compared with other classification models, the model accuracy rate is above 99.6%. The use of deep learning to classify URLs to identify Web visitors' intentions has important theoretical and scientific values for Web security research, providing new ideas for intelligent security detection.

INDEX TERMS Gated recurrent unit (GRU), malicious URL detection, network attack, character-level embedding, convolutional neural network, neural network model.

I. INTRODUCTION

Due to the rapid development of the Internet, cyber security has become an important research topic, and the energy waste caused by the occurrence of various cybersecurity incidents is immeasurable. In recent years, a large number of Internet companies have stolen user information data, resulting in the intrusion of users' online bank accounts. If the above information leakage incidents occur in the data platform of the relevant departments of the state finance and government affairs, the consequences will be unimaginable. The damage to national cybersecurity will be unprecedented. Web application layer attacks can cause long-term disruption to the resource availability, controllability, confidentiality, and integrity of data. Its influence is very persistent and secretive. A large number of web applications can construct executable commands, SQL injections, XSS and other web attacks simply by embedding executable code or malicious code in URLs. Therefore, the detection of malicious URLs has become the focus of intrusion detection.

Currently, there are also many scholars in the world who research the detection of malicious URLs. Most of the detection methods used are based on blacklists and rule detection [1]. Such detection methods can achieve high accuracy. However, they cannot be used to detect URLs that have not appeared. With the development of artificial intelligence, machine learning has been widely used in increasingly more aspects. However, machine learning-based attack detection models are based on the artificial selection of features [2] with data explosion and diversification. The determination of data features has become increasingly difficult, and machine learning-based detection models have been unable to respond adaptively to existing URL detection tasks. Deep learning is a subfield of machine learning. It can extract and learn features

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

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

from the most primitive input [3]. It eliminates the most time-consuming feature engineering in machine learning and can be more flexible in adapting to more complex attack behavior.

Detection of malicious URLs is an important topic in the field of network security. In deep learning, it can be regarded as a classification problem. We propose a CGRU neural network model specifically for network security. This is a multicategory method for URLs. It is a continuation of [4], and we follow the method of extracting malicious keywords and use the same data set. Specifically, we built a common URL-based dictionary and malicious keyword library and split the original URL into short strings for input. In the deep neural network model, the convolutional neural network is used to extract the features of the URL and effectively represent the features. Then, the GRU is used as a pooling layer to process the obtained feature sequences to obtain the hidden layer nodes. Finally, we use the softmax regression method for multiple classifications of URLs.

To evaluate this detection model, we chose 407, 212 different URLs as the training set. Malicious URLs included directory traversal, sensitive files, XSS, SQL injection, and other types of related Web attacks. We compared the performance of the CGRU neural network model and other deep learning methods in multiclassification in this paper. The experimental results show that the detection model proposed in this paper is very suitable for the detection of malicious URLs and can obtain 99.6% accuracy, and the training time is significantly better than other neural network models.

The structure of this paper is as follows. Section II describes the work related to the latest malicious URL detection methods, particularly how deep learning methods promote the development of malicious URL detection. Section III describes our proposed CGRU neural network classification model. Section IV specifically describes the malicious URL key character library based on Web attacks, the feature extraction method for original URLs. Section V presents the method we use to replace the pooling layer with GRU and the detailed description. In Section VI, we describe the datasets and methods used for the experiments and introduce the results of the experiments. Finally, the paper is summarized in Section VII and draws some conclusions.

II. RELATED WORK

In this section, we summarize some of the commonly used work to detect malicious URLs in the field of network security. We mainly describe blacklist-based detection methods, traditional machine learning methods and deep learning methods based on feature extraction.

A. BLACKLIST DETECTION AND MACHINE LEARNING METHODS

Research on the detection of malicious URLs can be divided into three main sections, using blacklists for detection, traditional machine learning methods, and deep learning based on feature extraction. For the blacklist detection method, Prakash *et al.* [1] used the Google browser to help establish their blacklist. URL detection was achieved by matching the IP address, hostname and directory structure. Sun *et al.* [5] built an automatic blacklist generator (AutoBLG) by adding prefilters to ensure that the blacklist is always valid.

Although the blacklist method can effectively find known attacks, this method is very fragile, and the blacklist update is slow and cannot be generalized to unknown URL recognition [6]. This year, various machine learning methods have been used for malicious URL detection. Ma *et al.* [7] used statistical methods to find the attributes of malicious URLs, extracted nine features for machine learning models, and compared the proposed classification model with naive Bayes, SVM models, and logistic regression experiments. Reference [8] based on heuristic and feature-based methods for feature extraction, an improved semi-supervised method is proposed to train the URL multiclassification model. This method has achieved better classification results.

B. DEEP LEARNING BASED ON FEATURE EXTRACTION

Machine learning methods have achieved effective results in the detection of malicious URLs [8], but manually extracting features is time-consuming and requires constant adjustment of features to accommodate changes in URLs in combination with human knowledge; this limits the accuracy of the classification model to some extent. In recent years, deep learning has been applied to intrusion detection and surpassed traditional detection methods. Bahnsen *et al.* [9] used different models for URL detection and found that recurrent neural networks do not need manually created features, and its detection results are better than random forest methods. J Saxe's work is most closely related to our research on character-level [10], [11] URL embedding [3]. This paper treats URLs, file paths, and registries as a short string and uses character-level embedding and a convolutional neural network to simultaneously extract features and classifications. It proposed the eXpose neural network and compared it with other models that extract features manually.

C. SUMMARY OF RELATED WORK

After reviewing existing techniques, we propose a new method to extract features more effectively and to detect malicious URLs. In addition to character-level embedding, we discussed the keyword library for malicious URLs and applied it to the feature extraction and proposed a detection model that combines a convolutional neural network and gated recurrent unit (GRU). After experimental verification, the proposed method performs well in improving the accuracy of malicious URL detection.

III. OVERVIEW OF THE PROPOSED MODEL

This section describes our proposed CGRU model for malicious URL detection. This model combines the characteristics of URLs in the field of Web attacks at the character level. It uses convolutional neural networks and gated recurrent units to extract features from URLs. Finally, it combines

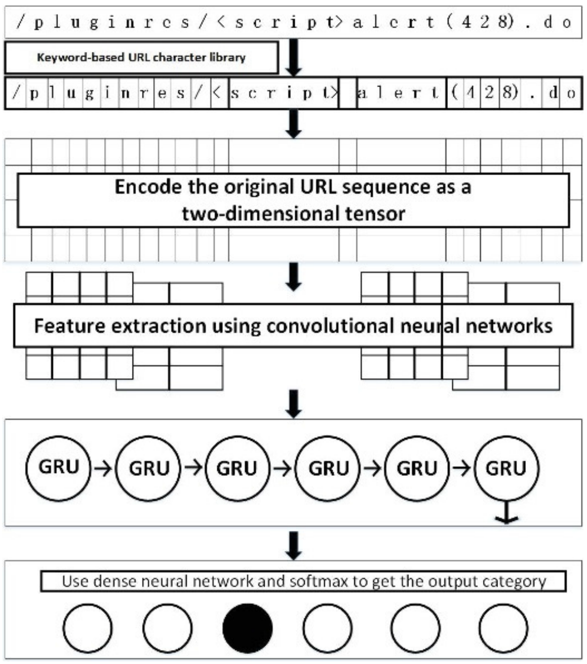
/pluginres/<script>alert(428).do
Keyword-based URL character library
/pluginres/<script>alert(428).do
Encode the original URL sequence as a two-dimensional tensor
Feature extraction using convolutional neural networks
GRU
GRU
GRU
GRU
GRU
GRU
Use dense neural network and softmax to get the output category

FIGURE 1. Keyword-based CGRU neural network classification model.

the classification module to detect Web attacks in URLs. Fig. 1 shows the overall structure of our neural network system. Our neural network model is divided into three parts:

1) Keyword-Based URL Character Embedding: The character embedding module is used to map the original URL character into a low-dimensional vector, thereby encoding the original sequence as a two-dimensional floating-point matrix. In the character embedding, the malicious keyword in the URL is distinguished from the ordinary character. Such differentiation can highlight the key part in the URL, which is advantageous in allowing the feature detection module to extract the representative feature more quickly.

2) Feature Extraction Module: The feature detection module uses the convolutional neural network to extract features on the abstract level of the URL and uses the GRU as a pooling layer, retaining the important features on the premise of preserving the context relationship. It uses a combination of different-length convolution windows to more fully extract features at each level. To make full use of the extracted features, the features extracted from the convolution windows need to be merged. The pooling section after the convolution operation uses the GRU as a pooling layer.

3) Classification Module: The fully connected neural network is used to classify the detected features. In our detection model, a stochastic gradient descent is used to jointly optimize the model. In terms of model evaluation indicators, in addition to the accuracy rate, precision, recall, and F1-score were selected to more fully evaluate our CGRU model.

The use of GRU as a pooling layer is an innovative part of the CGRU model. We have abandoned the traditional pooling methods and used GRUs to learn temporal characteristics. To achieve the effect of pooling, we only output the result of

TABLE 1. Malicious keywords.

URL sttack type	Malicious keyword
SQL Injection	and, or, xp_, substr, utl, benchmark, shutdown, hex, sqlmap, md5, hex, select, union, drop, delect, concat, orderby, exec
XSS Attack	script, iframe, eval, prompt, alert, javascript, cookie, onclick, Onerror, prompt
Sensitive File Attack	access_log, text/plain, phpinfo, proc/self/cmdline, /fckeditor/
Directory Travel	../, /, ..\, \
Normal	base64, wget, curl, redict, upload, ping, shal, java.lang

the last unit in all GRU units. This simplifies the characteristics while also taking into account the timing of the URLs.

IV. MALICIOUS URL KEYWORD LIBRARY AND FEATURE EXTRACTION METHOD

This section describes how to extract URL malicious keywords based on the characteristics of Web attacks and how to apply the established malicious keyword database to the feature extraction module of the original URL.

A. KEYWORD CHARACTER LIBRARY TO DETERMINE URL

We used the same training set as in [8]. The normal, SQL, XSS, sensitive files, directory traversals, and other types of URLs (numbers are kept equalized) were used as the initial tagged training set. Because our model uses the original length character sequence as input, this operation can be viewed as a dictionary query. The character embedding part was mapped to its corresponding vector using the letter and URL keyword of the printable English character and then embedded in the multidimensional feature space.

URLs are different from normal English strings. Characters in URLs are not completely independent. Malicious URLs have unique keywords, which are combinations of English characters, such as the '<script' keyword unique to XSS attacks. The method in [3] splits the URL into characters for processing and will lose part of the valid information. Based on this feature, in our implementation, we used an input vocabulary of 95 URL-valid characters and summarized the characters specific to the malicious URL, as shown in Table 1, and used it for character embedding. Our URL character library supports dynamic extensions with good adaptability to ensure that it can support more complex character processing in the future.

B. URL FEATURE EXTRACTION

We analyzed the characteristics of URLs of different attack types and applied their key features to the feature extraction part of deep learning. We used multicore convolutional neural networks to extract different levels of features through the use of different combinations of convolutional windows to ensure that the highest possible full-featured features [12], [13].

We used the original URL as the input and map each character as a word vector. We used a hash to map the constructed URL keyword character library to a list of integers, where each character corresponded to an integer. To make better use of context in characters, this article combined word embedding with the rest of the model. The original input character was converted to a low-dimensional vector, and these transformed vectors were spliced into a floating-point matrix and used as the input of the feature extraction part of the model. For the URL character library, we used the embedding layer provided by Keras to map the characters represented by the numbers into a 64-dimensional vector and then embedded the vectors into a 200 × 64 floating-point matrix. Compared to using the traditional one-hot method, the representation dimension of the feature was greatly reduced. We randomly initialized the character's embedded vector, and this embedded vector was trainable and back-propagated to the rest of the model. In the process of training, each embedded word vector was updated in the process of training the neural network, which made the embedded vector more semantic and more hyperplane separable.

When we obtained the two-dimensional tensor used to represent the input URL, we next needed to use a convolutional neural network to feature map the resulting vector. Convolutional neural networks use three ideas to help improve machine learning systems: sparse interaction, parameter sharing, and equivalence representation. This makes the convolutional neural network very suitable for processing data with local correlation. In our proposed detection model, we applied multicore convolutions and combined the convolution steps to detect local features of the input URL. This feature extraction method is similar to natural language processing [11], [14]. The convolution kernel uses a convolution operation on the embedding sequence of the original URL. The operation of each convolution window is as follows:

$$h_i = f(\omega_i x + b_i) \tag{1}$$

where ω_i is the ith convolution kernel of the convolutional layer, t is the length of the convolution window, b_i is the bias term corresponding to the ith convolution kernel, x is a matrix of the kth to $(k+t-1)$th character vectors in the embedded character vector, and $f(\cdot)$ represents the activation function of the convolutional layer. In this paper, we selected ReLU and obtained a new feature h_i after convolution. The literature [12] describes in detail that different convolution window combinations can extract features at different levels of abstraction. Therefore, we also conducted comparative experiments on the values of the convolution window in the experimental results section. Once the feature extraction was performed using different convolution windows, a convolution layer was followed by a merge layer that was used to merge features extracted from different convolution windows. The merging layer refers to splicing that ensures that all extracted features are fully preserved.

In the feature extraction section, we used multiple convolution kernels to generate feature maps. Each convolution kernel had the same length. For the embedded character vector $x_{k:k+t-1}$, the new features generated by n convolution kernels are expressed as follows:

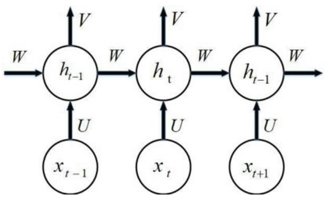

FIGURE 2. RNN internal status.

$$W = [h_1, h_2, \cdots, h_n] \tag{2}$$

Here, we obtained the higher-order feature representation of the embedded characters, and the resulting features were fed into the GRU instead of the pooling layer.

After the convolution operation, we did not obtain the features directly for the classification operation. Because of the large number of data features, a large number of training parameters were required, and it was extremely easy to overfit. Therefore, the pooling layer was used for further sampling of features to obtain more representative features. The commonly used pooling operations are max-pooling, mean-pooling, and k-max pooling. [3] proposed a SumPool method to reduce the tensor dimension. Once the maximum pooled or averaged pooling was used to obtain the higher order features obtained by the convolution, the feature sequence information in the feature was destroyed. In neural networks that classify images, the maximum convolution guarantees translation invariance, and the average pooling fully considers all high-order features in the space. At this point, if you want to obtain further timing characteristics, you will generally continue to merge the LSTM [15] layer or the RNN layer after the pooling layer, but this network structure is more complex than our proposed CGRU model and will create many parameters and a training time burden. Therefore, for the URL, we use GRU instead of the traditional pooling operation to better obtain the timing-based representative features on higher-order features.

To prove that character-level neural networks can extract richer features, we used the 21-dimensional feature vectors extracted in [4] as comparison experiments. The detailed experimental setup is described in Section VI. From the experimental results, we can see that our CGRU model is superior to artificially extracted features in terms of detection indicators.

V. GATED RECURRENT UNIT-BASED TIMING POOLING

RNN is a neural network that models sequence data. The current output a of a sequence is not only related to the input b at the current time but is also affected by the output c of the node at the previous time. The internal state transition of the RNN is shown in Fig. 2. This neural network can be viewed as a dynamic system with more dynamic characteristics than the feed-forward neural network.

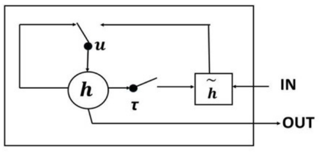

FIGURE 3. GRU structure.

Theoretically speaking, RNN can combine historical hidden state vectors to calculate the current output. However, the standard RNN has problems such as vanishing gradient and exploding gradient [13], [14] when it is used, and it cannot learn long-term dependence. To solve these problems, long short-term memory (LSTM) [13], [15] emerged. That is, in the traditional RNN structure, a cell unit for controlling input, forgetting, and output is added so that the weight of the self-looping depends on the context. To avoid the problem of gradient disappearing or gradient explosion, LSTM contains three thresholds, resulting in its internal structure being very complex. The GRU (gated recurrent unit) was proposed by Cho *et al.* [16] in 2014. Compared to the three complex thresholds of the LSTM, the GRU has only two doors, i.e., an update gate and a reset gate. While maintaining the LSTM effect while also streamlining the structure, the GRU unit is shown in Fig. 3.

The main idea of our CGRU model is based on the GRU unit implementation. Among them, the reset and update gates in the GRU can independently "ignore" part of the state vector; that is, a single gated unit controls both the forgetting factor and the updating state unit. Compared to the standard RNN, an update gate and a reset gate are added. The definitions of the update gate, reset gate, candidate activation state, and activation state in the CGRU model proposed in this paper are as follows:

$$u_t = \sigma(W_u \cdot [h_{t-1}, c_t]) \tag{3}$$

$$\tau_t = \sigma(W_\tau \cdot [h_{t-1}, c_t]) \tag{4}$$

$$\tilde{h}_t = \tanh(W \cdot [\tau_t \cdot h_{t-1}, c_t]) \tag{5}$$

$$h_t = u_t \cdot \tilde{h}_t + (1 - u_t) \cdot h_{t-1} \tag{6}$$

where $c_t \in W$ and $W = [c_1, c_2, \cdots, c_n]$ are high-order features extracted from the convolutional layer. To capture the temporal and long-term dependencies of URLs, we select the last step GRU from the n high-order features as a representation of the final higher-order features of the URL, and the result of the last GRU is sent to the final classification layer of the neural network model. The GRU can retain important features through the update gate and reset gate, ensuring that the contextual relationships in the high-order features obtained by processing the upper convolutional neural network are not lost. The GRU further obtains representative time-series-based features from higher-order features, which have been applied in [11], and uses LSTM to achieve satisfactory results in text classification problems. We used GRU instead of LSTM and conducted comparative experiments in the subsequent experimental section.

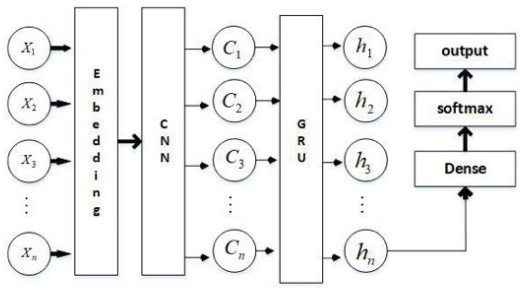

FIGURE 4. CGRU model using GRU instead of pooling layer.

TABLE 2. Data distribution of URL dataset.

URL type	URL number
Normal	65767
SQL Injection	48733
XSS Attack	43322
Sensitive File Attack	110119
Directory Travel	57563
Other Attacks	81708
Total	407212

Here, multiple GRU units function as a pooling layer. From the perspective of natural language processing, this can also be seen as a combination of different neural networks in text processing [17]. Once the features detected by the convolutional neural network [18], [19] are selected again, the parameters of the operation can be reduced and more accurate features can be obtained. After the high-order features are obtained by the GRU layer, a standard fully connected neural network is used for multiple classifications of URLs. The hybrid model with GRU units is shown in Fig. 4. To prevent over-fitting, set the dropout to 0.5 and use multiple categories of cross-entropy as the loss function. Also. use softmax regression to obtain the final class output. The overall model is optimized using a stochastic gradient descent algorithm.

VI. EXPERIMENTAL STUDIES

We combined some of the datasets in [4] and the datasets we collected as training datasets for experiments. Most of the data originated from a well-known Chinese Internet security company. We cleaned and deduplicated the datasets and finally obtained a more balanced data distribution based on ModSecurity's rule-based modules and manual annotations. Including more than 65, 000 normal URLs and 340, 000 malicious URLs, malicious URLs were classified into different attack types. The specific data distribution is shown in Table 2. According to the type sampling method, the dataset was divided into a training set, verification set and test set according to a ratio of approximately 3:1:1. Our experiments were performed on this dataset.

A. EXPERIMENTAL SETTINGS

Our model used a fixed-length URL as input. Based on the collection of URL keywords and statistics on URL length in the dataset, we used 200 as the maximum length of the

TABLE 3. The parameter settings of our CGRU model.

Parameter	Settings
Word Embedding Dimension	128
Convolution Kernel Size	{2*128, 4*128}
Convolution Kernel Numbers	{128, 128}
Activation Function	RELU
GRU Dimension	64
Dense Dimension	6
Dropout Rate	0.5
Initial Learning Rate	0.01
Learning Rate Drop Rate	0.5
Batch Size	100
Optimizer	SGD Optimizer

training set of strings. For URLs that were less than the maximum length, we used special characters for padding, and for URLs that were longer than the maximum length, we cut off the beginning of the string.

Our model is implemented in Python 3.5 using Keras v2.2. At the embedding level, we used a uniform distribution to initialize the character vector in the URL and train the character vector along with the rest of the model during training so that the character vector could learn the semantics. The dimension of the character vector was set to 128, which is a trade-off between precision and computational complexity. In the convolutional layer, we used the multi-core convolution method and set the convolution kernel to 256, and to ensure sufficient learning characteristics, we used a combination of convolution window k, where $k = \{2, 4\}$. The CNN activation method was set to the ReLU function. Because we used two types of convolutional neural networks, to fully integrate the results, we needed to use the merge layer provided by Keras to merge the merged results. The number of nodes in the GRU layer was 64. Because the GRU was used here to extract timing-based features from the higher-order features acquired by the merge layer to achieve the pooling effect, only the last output of the output sequence was returned. The output dimension was set to 6, which is the number of URL categories in the full connection layer, and softmax regression was used to obtain the output category.

In our training, we set the batch size to 100 and the number of training rounds to 30, and we used a small batch of random gradient descent to train the model. The loss function used was the cross-entropy loss function.The optimizer is a stochastic gradient descent optimizer with an initial learning rate set to 0.01, the learning rate has a drop rate of 0.5, and the learning rate is dropped every 5 epoch. And the learning rate is decayed during the training process.The parameter settings of our CGRU network is described in Table 3.

B. CGRU MODEL EXPERIMENT RESULTS

The evaluation index of the CGRU model on the validation set is shown in Table 4. We chose the best result of the CGRU model using different convolution windows for experiments. The combined convolution window size was {2, 4}. Our CGRU achieved an accuracy of over 99.6% and achieved high results in accuracy, recall and F1 values. This shows that our model has a strong generalization and can accurately detect the vast majority of malicious URLs.

TABLE 4. Results of the CGRU model on the data set (%).

	Accuracy	Precision	Recall	F1-Score
CGRU	99.61	99.63	99.58	99.61

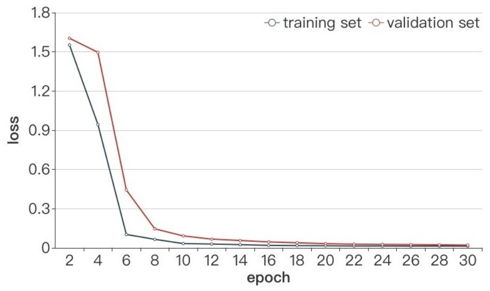

FIGURE 5. Loss curve of the loss function of the CGRU model in the training set and validation set.

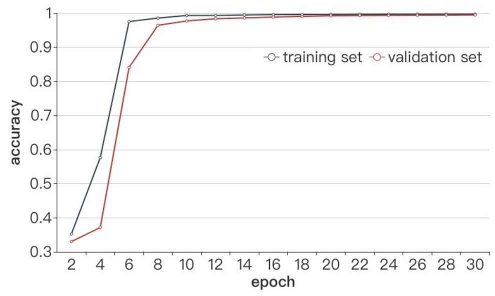

FIGURE 6. Accuracy curve of the CGRU model on the training set and validation set.

Fig. 5 shows the change in the loss function in the training set and verification set within 30 epochs of training in our CGRU model. Fig. 6 shows the changes in the accuracy of the training set and verification set in the CGRU training in 30 epochs. It can be seen from the figure that for the training set and verification set, the model converges normally.

For the data in the test set, we calculated the actual results of the CGRU model detection, that is, the model output confusion matrix; the results are shown in Table 5. It is seen that our model detected almost all directory traversal attacks and detected most of the other types of attacks. In the detection of normal access URLs, a small portion of the URLs was mistaken for other attack types and sensitive file attacks. This may be because this small part of the URL contained more sensitive characters. However, these false positives were insignificant compared to the successful detection of normal URLs. Therefore, overall, our CGRU model was very successful, and because of its high performance, it can be put to practical use.

C. EFFECTS OF CONVOLUTION WINDOW SIZE ON CLASSIFICATION

We selected different convolution windows for comparison experiments. From the perspective of natural language processing, the length of each word's mapping should not exceed 5. For our URL data, after statistics, the length of the

TABLE 5. CGRU model results on the test set.

Predicted class	Actual class		
	Directory travel	Other attack	Sensitive file attack
Directory travel	9,900	0	9
Other attack	3	15,967	4
Sensitive file attack	9	0	21,963
SQL injection	2	7	1
XSS attack	24	0	14
Normal	9	95	45

Predicted class k	Actual class		
	SQL injection	XSS attack	Normal
Directory travel	0	0	1
Other attack	2	1	23
Sensitive file attack	5	5	18
SQL injection	9,682	6	2
XSS attack	8	8,536	18
Normal	4	2	10,612

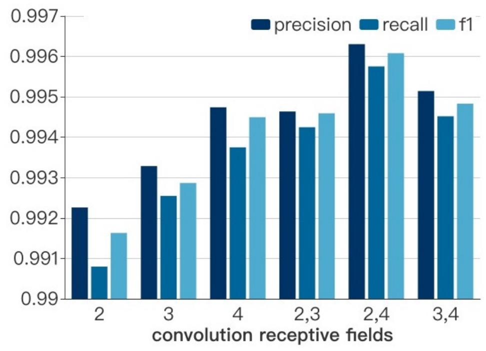

FIGURE 7. The effect of different convolution windows on the results.

convolution window was chosen to be less than 5. As shown in Fig. 7, we can see that when only a fixed-length convolution window is used, when the size of the convolution window is less than 5, the larger the length of the convolution window, the better the performance of the model.In order to further improve the evaluation indicators, we will combine the convolution windows.When the size of the combined convolution window is $\{2, 4\}$, it means that there are two convolution layers for feature extraction,the convolution window of one convolutional layer has a size of 2, and the convolution window of another convolutional layer has a size of 4.We use the merge layer provided by Keras to merge the features extracted by the two convolutional layers and input them to the next layer of the neural network model.

We performed a comparison between the fixed window length and the combined convolution window length in terms of precision, recall and F1-score. Because the combined convolution window extracts features at different levels, it generally performed better than the fixed window length.

D. EFFECTS OF FEATURES ON CLASSIFICATION

We have described character-level feature extraction methods based on keywords and referred to the 21-dimensional features described in [4], as shown in Table 6. To compare the experimental results, we used artificially extracted datasets for the classification of models.

TABLE 6. The 21-dimensional features extracted in [4].

Feature type	Feature name
The length of features	URI length
	URI maximum length
	URI average length
	Parameter length
	Parameter average length
	Name maximum length
The number of features	URI number
	Parameter number
The type of features	URI type
	Name type
	Value type
The risk level of features	URI risk level
	Name risk level
	Value risk level
Other advanced features	Digit percentage
	Letter percentage
	URL unknown amount
	If contain keywords
	Nginx test
	Value_contain_ip

Because most of our experimental datasets came from the datasets in [4], we used the features extracted in [4]. The features here include length features, quantity features, type features, hazard classes, and other features. After a comprehensive analysis from multiple dimensions, feature sets that can discriminate between abnormalities and non-abnormalities were extracted.

We performed 21-dimensional feature extraction of the used data set and then sent the extracted features to our CGRU model. Because the data that can be directly sent to the model were obtained here, the embedding layer suitable for the character level was removed from the CGRU model, and the rest of the parameter updating methods and optimization methods were the same as the original CGRU model. The detailed experimental comparison results are shown in Fig. 8.

As shown in Fig. 8, when we used artificially specified 21-dimensional features as a classification basis, performance results were significantly inferior to the use of character-level inputs and the use of neural networks for feature extraction. Moreover, when feature extraction was performed using character-level data embedding, the convergence of the model was significantly earlier than the extraction of human features, and the learning curve was smoother.

It is undeniable that experts rely on analysis and experience to give more comprehensive characteristics and better results. However, it is often impossible to judge and extract features of an unknown attack. This requires less concern when using neural networks for feature extraction. We only need to send all the input data into the model in the form of character level. The model will perform a series of mathematical operations through the neural network to extract the most suitable features for the classification of the data set.

E. RESULTS FROM DIFFERENT MODELS

In this experiment, the CGRU model using the text embedding method containing malicious keywords proposed in this paper was compared with other neural network models

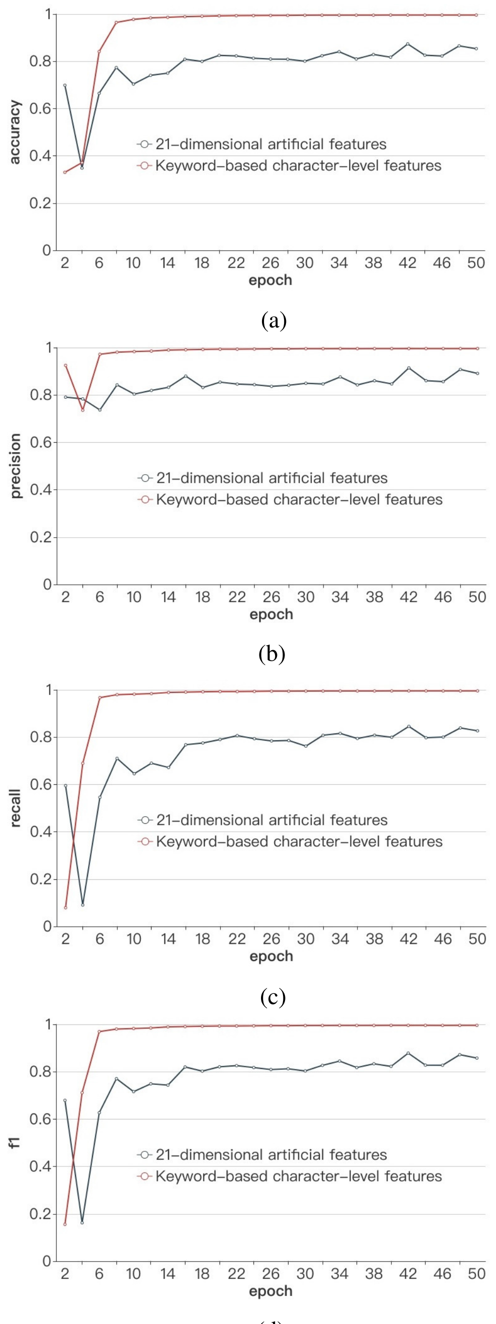

FIGURE 8. **The effect of different characteristics on the result. (a) Accuracy. (b) Precision. (c) Recall. (d) Fl-score.**

or machine learning models using characters for text representation. All models used the dataset shown in Table 2. We detect malicious URLs from the perspective of text processing, construct a specific combination of CNN and GRU, and obtain the final detection model to achieve network security monitoring. This idea is similar to the idea of neural network application and network traffic classifier (NTC) problem in [20].

We selected a single model and a mixed model. In the mixed control model, we used the model "eXpose" proposed in [3], which uses 4 convolutional neural networks and uses 4 fully connected neural networks. We also used the C-LSTM proposed in [11], which uses a combination of long- and short-term memory networks and convolutional neural networks for feature extraction. Two models were used in a single model control; One is "Char-CNN", which performs Word Embedding on input data, uses convolutional neural networks to extract features, and uses a fully connected layer for classification.The other is "Char-LSTM", which performs Word Embedding on the input data, uses LSTM Networks to extract feature features in the time dimension, and uses the fully-connected layer for classification. The comparison results of the above models on the test set are shown in Table 7.

TABLE 7. **Comparison of CGRU model with other classification models.**

Model	Accuracy	Precision	Recall	F1-score
Char-CNN	98.58	98.58	98.55	98.56
Char-LSTM	97.42	97.29	96.41	96.85
eXpose [3]	99.46	99.47	99.41	99.44
C-LSTM [11]	99.13	98.94	98.75	98.84
CGRU	99.61	99.63	99.58	99.61

To fully demonstrate the capabilities of the different models, in the optimization aspect, dropout = 0.5 was set for each model, and for the accuracy on the validation set, if there was no change in 5 rounds, the strategy of stopping prematurely prevented over-fitting. To speed up the convergence, batch standardization was used to normalize the activation value of the previous layer on each batch. The initial learning rate was set at 0.01 in terms of the learning rate, and then the learning rate was decremented every 5 epochs. Due to our dataset, the distribution of each item was not very balanced. Therefore, in terms of evaluation criteria, it includes not only the accuracy rate but also the accuracy rate, recall rate, and F1 value.

We recorded the performance of these models during the training process, as shown in Fig. 9. After optimization, each model achieved high accuracy.Among them, the "Char-CNN" model achieved high accuracy at the beginning of training, but as the training progressed, the accuracy rate did not increase much.For "Char-LSTM", the accuracy of the model is lower than other models, and the accuracy of the model proposed in [11] is lower than the "eXpose" model proposed in [3] and our CGRU model.The model proposed in [3] has achieved high accuracy in the beginning, and the final accuracy is very close to our model, but the model proposed in [3] is complex in structure and requires high computational resources.

As shown in Table 7, the performance of the single model was slightly inferior to that of the hybrid model because the hybrid model fully extracts multiple levels of features. Among them, the model proposed in [3] also achieved higher results, but there was a large difference in the complexity of the structure between this model and our CGRU model. Reference [3] used multiple convolutional neural networks for feature extraction and used three fully connected layers

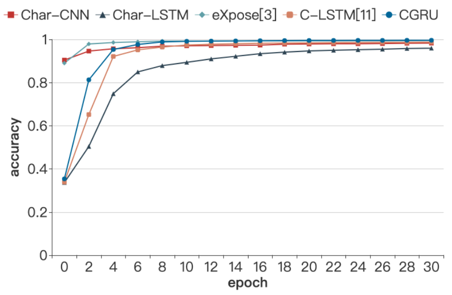

FIGURE 9. **Accuracy curve of the different model on the training set.**

TABLE 8. **Time and required parameters between CHRU and eXpose [3] on the test set.**

	eXpose [3]	CGRU	C-LSTM [11]
Time (s)	1054.63	640.78	302.55
Total params	1,718,150	217,222	96,134

for classification, so the model is inferior to our CGRU model in both memory consumption and training time. Moreover, due to the use of a combined convolution window and the use of GRUs for pooling to achieve fuller features, our model can achieve higher evaluation index performance. The classification model used in [11] was the closest to our model structure. It used LSTM as the pooling layer to extract high-order features. Moreover, the feature extraction of embedded characters is not performed using a combination of convolution windows of different lengths. Therefore, the structure of the model is relatively simple. For more than 60,000 pieces of data in the test set, the length of the processing required for the two models and the required parameters of the model are shown in Table 8.

From the classification duration and required parameters, our model consumed much less memory than eXpose. This is because eXpose uses four convolutional neural networks for fusion and uses three fully connected neural networks for classification. We found that the C-LSTM model requires the shortest calculation time and the minimum required calculation parameters. At this point, our proposed CGRU model is inferior. However, as seen in Table 7, the CGRU is superior to the model proposed in [11] in terms of various evaluation indicators. Our model uses a two-window convolutional neural network for fusion and taking the output of the last block of the GRU as an extracted high-order feature; it greatly simplifies the model while ensuring the temporal and long-term dependencies of the feature.

VII. CONCLUSION

This paper introduces a neural network model CGRU for malicious URL detection in the field of cyber security. Through comparison experiments with the feature of manually extracting malicious URLs and comparison experiments with other classification models, we proved that our model has a good effect of malicious URL detection.

The innovations of our proposed model are mainly manifested in two parts. In the data pre-processing section, our model does not use traditional artificial features but adopts a method of directly extracting features from the original input; this approach greatly ensures the pertinence and effectiveness of features. In the design part of the model, compared to the traditional pooling operation after convolution, we innovatively use GRU for pooling, which ensures the timeliness of higher-order features while streamlining the parameters required for training.

Through the analysis of the experimental results, we can see that our model has performed well enough in terms of detection. In terms of energy, our network security detection model can effectively prevent improper use and the waste of information resources and can optimize network resources and computing resources to a certain extent. In the future, we will conduct optimization research to reduce memory consumption while ensuring excellent test results. At the same time, we can study how the model is updated online to ensure that the model has a greater advantage in actual network detection.

REFERENCES

[1] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: Predictive blacklisting to detect phishing attacks," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–5.

[2] D. Sahoo, C. Liu, and S. C. H. Hoi. (2017). "Malicious URL detection using machine learning: A survey." [Online]. Available: https://arxiv.org/abs/1701.07179

[3] J. Saxe and K. Berlin. (2017). "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys." [Online]. Available: https://arxiv.org/abs/1702.08568

[4] B. Cui, S. He, X. Yao, and P. Shi, "Malicious URL detection with feature extraction based on machine learning," *Int. J. High Perform. Comput. Netw.*, vol. 12, no. 2, pp. 166–178, 2018, doi: 10.1504/IJHPCN.2018.10015545.

[5] 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.

[6] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Proc. 6th Conf. Email Anti-Spam (CEAS)*, Sacramento, CA, USA, 2009, pp. 59–78.

[7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Paris, France, Jun./Jul. 2009, pp. 1245–1254.

[8] J. Yang, P. Yang, X. Jin, and Q. Ma, "Multi-classification for malicious URL based on improved semi-supervised algorithm," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE)*, vol. 1, Jul. 2017, pp. 143–150.

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

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.

[11] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau. (2015). "A C-LSTM neural network for text classification." [Online]. Available: https://arxiv.org/abs/1511.08630

[12] Y. Kim. (2014). "Convolutional neural networks for sentence classification." [Online]. Available: https://arxiv.org/abs/1408.5882

[13] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[14] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2012, vol. 52, no. 3, p. 1310–1318.

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

[16] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. (2014). ''On the properties of neural machine translation: Encoder-decoder approaches.'' [Online]. Available: https://arxiv.org/abs/1409.1259

[17] Y. Fan, L. Gongshen, M. Kui, and S. Zhaoying, ''Neural feedback text clustering with BiLSTM-CNN-Kmeans,'' *IEEE Access*, vol. 6, pp. 57460–57469, 2018.

[18] A. Conneau, H. Schwenk, Y. Lecun, and L. Barrault. (2016). ''Very deep convolutional networks for natural language processing.'' [Online]. Available: https://arxiv.org/abs/1606.01781v1

[19] Y. Zhang and B. Wallace. (2015). ''A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.'' [Online]. Available: https://arxiv.org/abs/1510.03820

[20] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, ''Network traffic classifier with convolutional and recurrent neural networks for Internet of Things,'' *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

WENCHUAN YANG received the B.E. and M.S. degrees from Sichuan University, Chengdu, China, in 1994, and the Ph.D. degree from Peking University, Beijing, China, in 1997, all in computer science. He is currently an Associate Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing. His research interests include data science and business intelligence.

WEN ZUO received the B.E. degree in computer science from Xidian University, Xi'an, Shanxi, China, in 2012. She is currently pursuing the M.S. degree with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include big data technology, artificial intelligence, and network security.

BAOJIANG CUI received the B.S. degree from the Hebei University of Technology, Tianjin, China, in 1994, the M.S. degree from the Harbin Institute of Technology, Harbin, China, in 1998, and the Ph.D. degree in control theory and control engineering from Nankai University, Tianjin, in 2014. He is currently a Professor with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include the detection of software, cloud computing, and big data.