applied sciences

MDPI

Article

Classification of Malicious URLs by CNN Model Based on Genetic Algorithm

Tiefeng Wu, Yunfang Xi *, Miao Wang and Zhichao Zhao

School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266520, China
* Correspondence: xi_yunfang@163.com

Abstract: Researchers have proposed many models for the identification of malicious URLs in network security, but they have not achieved good results. In order to improve this defect, the current popular machine learning algorithm is combined to train the model, thus improving the accuracy of malicious URL classification. This paper proposes a model of a convolutional neural network based on genetic algorithm optimization. Firstly, the genetic algorithm was used to reduce the data dimension of the grammatical features, structural features, and probabilistic features in the extracted malicious URL text, and then the convolutional neural network was used to establish the model and classify the malicious URL. Through experimental verification, the model has achieved good results. Compared with the traditional machine learning model, it improves the accuracy of malicious URL recognition and provides a reference for malicious URL recognition.

Keywords: malicious URL; dimension reduction; convolutional neural network; classification

Citation: Wu, T.; Xi, Y.; Wang, M.; Zhao, Z. Classification of Malicious URLs by CNN Model Based on Genetic Algorithm. *Appl. Sci.* **2022**, *12*, 12030. https://doi.org/10.3390/app122312030

Academic Editor: Aleksander Mendyk

Received: 21 October 2022
Accepted: 22 November 2022
Published: 24 November 2022

1. Introduction

At present, the global network security problem is constantly updated with the development of the economy, and the way of a network attack is also escalating. The network security market will gradually recover in 2021. The '2021 China Network Security Report' released by Ruixing Company points out that it makes a detailed analysis of malware, malicious URLs, mobile security, enterprise security, and other fields. The report points out that network attacks are becoming more and more frequent, and maintaining network security and defending against network attacks have become the top priority. At present, many malicious URLs are mixed into the URLs accessed by users in their daily network activities. With the continuous updating of technology, malicious URLs are very similar to benign URLs, so it is particularly important to identify malicious URLs correctly and efficiently.

Researchers use a variety of benchmark models for malicious URL detection. Abdulhamit Subasi et al. [1] compared the accuracy of KNN, SVM, CART, C4.5, LADTree, and NBTree using the data set of URL instances and verified that the classification results of support vector machine and KNN are more accurate through integration technology. Anjali B. Sayamber et al. [2] proposed a method for the automatic classification and detection of malicious URLs using a naive Bayesian classifier. For a wide range of benchmark data sets, the naive Bayesian model using probabilistic model learning has better accuracy than the support vector machine model. Liu Jian et al. [3] proposed a multi-level filtering detection model using a machine learning algorithm to identify malicious URLs. By training the key threshold of the classifier, the classifier only determines the URLs they are good at, giving full play to their own advantages. If these classifiers are not good at classifying a URL, then use multiple classifiers to vote. Finally, compared with naive Bayes, decision tree, and SVM model, this method improves the accuracy of detecting malicious URLs. Vara Vundavalli [4] used logistic regression, neural networks, and different types of naive Bayes algorithms to classify malicious URLs, and finally, naive Bayes obtained better results. Sheikh Shah Mohammad Motiur Rahman et al. [5] evaluated the performance of various machine learning

classifiers by AUC-ROC curve, accuracy, misclassification rate and mean absolute error for the identification of phishing URLs. The best AUC area was obtained from the random forest and multi-layer perceptron, respectively, and the accuracy of stacking generalization in binary classification and multi-class feature sets was higher. Li Zeyu et al. [6] studied the effect of various machine learning models, especially ensemble learning models, on malicious URL recognition. Through a number of indicators such as recall rate, accuracy rate, and AUC value, it is verified that the inheritance learning method is superior to the traditional machine learning model, and the random forest performs best. Thuy Thi Thanh Pham et al. [7] used three different deep neural networks, CNN, LSTM, and CNN-LSTM, to detect malicious URLs, but did not compare the hidden layer and the number of neurons in the experiment to find the optimal network parameters. Chen Z et al. [8] proposed an improved multi-layer recursive convolutional neural network model based on the YOLO algorithm to detect malicious URLs. Compared with Text-RCNN, BRNN, and other models, this method has higher accuracy. In the experiment, the truncation method is used to standardize the length of all URLs. It is inevitable to lose some information when facing longer URLs. On the basis of the benchmark machine learning model, the recognition rate of malicious URLs can also be improved by improving the feature engineering method. Tie Li et al. [9] proposed a feature engineering method combining linear and nonlinear spatial transformation. New features are generated by five spatial transformation models and applied to the classifier, which significantly improves the recognition rate of KNN, linear support vector machine, and multi-layer perceptron.

In the current research, most of the detection methods of malicious URLs extract features from the content of URLs. Kumi S et al. [10] combined classification with association rules and proposed a data mining algorithm based on association classification, which uses URLs and web content features to detect malicious URLs. A. Saleem Raja et al. [11] proposed a weighted method that only contains URL lexical features, extracts a small number of features for malicious URL detection, and measures machine learning algorithms from the perspective of performance and execution time. Random forest and KNN algorithms obtain good results. Apoorva Joshi et al. [12] extracted static lexical features from URL strings, used the ensemble classification method learned by machine learning to detect them, and concluded that the pure lexical method could generate fast real-time determination of URLs in lightweight systems. Chen Kang et al. [13] proposed a detection method based entirely on lexical features. The convolutional neural network extracts and classifies the URL strings and finally obtains a better classification result. Yuan J T et al. [14] proposed a joint neural network algorithm model of Bi-IndRNN and CapsNet to identify and detect malicious URLs, extract vector features and texture features at the character and word levels, and perform feature fusion. Classification by a joint neural network effectively improves the efficiency and accuracy of malicious URL detection. H Le et al. [15] proposed a deep neural network based on CNN for URLNet for malicious URL detection. For manual feature engineering and the inability to handle features that are not visible in the test URL, the characters cnn and Word cnn are proposed, and the network is jointly optimized.

In addition, many researchers also identify URLs from the aspects of extracting texture features and structural features. Yuan J et al. [16] proposed a parallel neural joint model algorithm for the analysis and detection of malicious URLs, using CapsNet to combine texture features with text features and capture multi-modal vectors. Zhao Gang et al. [17] proposed a decision tree intelligent detection method for two-dimensional code URLs. The accuracy rate in identifying regular URLs and two-dimensional code malicious URLs has a relatively good effect, but the selected data volume is low, and the model's generic ability is not well trained. Liu C et al. [18] proposed a statistical method to study the character features of malicious URLs and obtain the character distribution features and structural features. The extracted features were combined with random forests to obtain better performance than the original features, and the parameters of random forests were adjusted to be the best. Lin Helen et al. [19] proposed an efficient method for detecting

malicious URLs based on segment pattern, which parses the three semantic segments of the domain name, path name, and file name in the marked malicious URL and quickly calculates the pattern of each semantic segment of the malicious URL through the inverted index of the triple as the term, and finally determines the segment pattern according to the inverted index. There is no research on other features, such as IP address for the content of domain name, path, and file name. Gabriel AD et al. [20] proposed a system for analyzing URLs in network traffic. Each correctly classified URL is reused as part of a new data set. Different clustering techniques are used to identify missing features on malicious URLs. Through the OSC perceptron optimization algorithm, the final recognition accuracy is improved, and the overall detection rate is increased to 82.9%.

Based on the above research background, in order to better identify malicious URLs and improve the accuracy of detection, there are some unobvious features in the URL text. Based on the grammatical features, structural features, and probability features of URLs, a convolutional neural network model based on genetic algorithm optimization is proposed. High-dimensional feature space, on the one hand, will consume a lot of training time; on the other hand, there may be some interference information on the training results. Therefore, firstly, the genetic algorithm is used to reduce the dimension of URL features, and the feature subset with a good classification effect is searched. Secondly, the feature subset is classified by a convolutional neural network. Finally, compared with the traditional machine learning method, the accuracy of malicious URL recognition is improved.

2. Materials and Methods

2.1. Feature Extraction

In this paper, the URL was analyzed and processed. Firstly, the data were preprocessed. By observing the content contained in the URL and comparing the malicious URL with the benign URL, the initial feature data set was obtained. In order to further improve the efficiency of data processing, the feature vector is reduced by the feature extraction method of the genetic algorithm.

Genetic Algorithm (GA) is a computer simulation study of biological systems. It is a stochastic global search and optimization method that imitates the mechanism of natural evolution. Its essence is an efficient global search method with parallel ability, which can search the whole solution quickly without falling into the trap of local optimum. At the same time, the genetic algorithm does not have the limitation of derivation and function continuity. The probabilistic optimization method can automatically obtain and guide the optimized search space, adaptively adjust the search direction, and does not need to determine the rules.

The feature extraction process of the genetic algorithm is shown in Figure 1. Firstly, the features were encoded and the population was initialized, and the fitness of the individuals corresponding to each chromosome was evaluated. Following the principle that the greater the fitness, the greater the probability of being selected, two individuals were selected from the population as parents, and the crossover operation was performed to generate offspring, and then the chromosomes of the offspring were mutated. Repeat the above steps until a new population is generated.

2.1.1. Population Individual Coding

The population and encode the individuals of the population are randomly initialized by binary coding. Feature extraction is to extract d features (d < D) from D features and encode all features into D strings composed of 0 and 1. The selected features are 1, and the unselected features are 0. For example, a chromosome g = [0, 0, 1, 0, 1, 0,..., 0] represents the selection of the third and fifth characteristics.

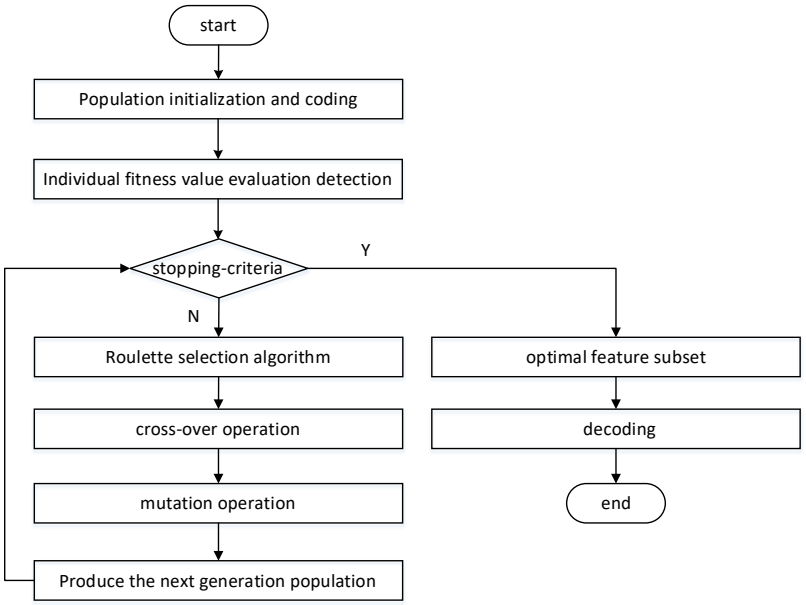

Figure 1. Genetic algorithm feature extraction flow chart.

2.1.2. Calculation of Fitness Function

The result of feature selection based on a genetic algorithm mainly depends on the definition of the fitness function. This paper uses the accuracy of support vector machine (SVM) classification as the fitness function. The selected features are directly classified by the support vector machine algorithm, and the accuracy rate is obtained. The accuracy rate is used to determine whether it is the optimal feature. The support vector machine is based on the principle of structural risk minimization, which has strong generalization ability and global optimality. Let $\varphi(x)$ be the vector of the sample x after mapping, and divide the hyperplane into Equation (1).

$$f(x) = w^T \varphi(x) + b \tag{1}$$

In Equation (1), w is the weight vector and b is the offset. Transform the problem into a quadratic programming problem,

$$\min\left[\frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n}\xi_i\right] \tag{2}$$

$$s.t.y_i[(w \cdot \varphi(x_i)) + b] - 1 + \xi_i \geq 0 \tag{3}$$

where $\xi_i \geq 0$, ξ_i is a relaxation variable, and the data x_i can be correctly classified when $0 < \xi_i < 1$; when $\xi_i \geq 1$, data x_i are misclassified. $i = 1, 2, \cdots, n$, C is the penalty parameter. The dual form of the problem is,

$$\max\left[\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(\varphi(x_i) \cdot \varphi(x_j))\right] \tag{4}$$

$$s.t.\sum_{i=1}^{n}\alpha_i y_i = 0, 0 \leq \alpha_i \leq c, i = 1, 2, \cdots, n \tag{5}$$

Here, α_i is the Lagrange multiplier and y_i is the target value. According to the KKT condition in optimization theory, only a small number of sample values are not zero. Solving nonlinear transformation needs to define kernel function. In this paper, the Gaussian kernel is used:

$$k(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2) \tag{6}$$

where γ is the kernel parameter. When the γ value is too large or too small, it will produce too-small or too-large training errors. Finally, the accuracy of the classification results as a fitness function.

2.1.3. Selection Operator

The selection operation is used to determine which individuals are selected from the parent group as parents. Through the evaluation of individual fitness, the higher-fitness individuals are selected, and the lower-fitness individuals are eliminated. In this paper, roulette is used. The probability of each individual entering the next generation is the proportion of its fitness value in the sum of individual fitness values in the whole population. Assuming that the population number is M and the fitness of individual i is, the probability that individual i is selected is

$$P_i = \frac{fitness_i}{\sum_{k=1}^{M} fitness_k} \tag{7}$$

2.1.4. Crossover Operator and Mutation Operator

The function of crossover is to inherit the excellent genes of the parent population to the filial generation to ensure the search ability and stability of the genetic algorithm. The crossover operator used in this paper is a single-point crossover. Two chromosomes are selected, and the genetic genes of the two parts are segmented and exchanged at random positions to obtain two different offspring chromosomes.

As shown in Figure 2, P1 and P2 on the left represent two individuals of the parent. A point is randomly selected from the two parent chromosomes (the part indicated by the arrow in the figure) as a crossover breakpoint, and the part after the point is exchanged to obtain two different offspring chromosomes C1 and C2, on the right side of the figure.

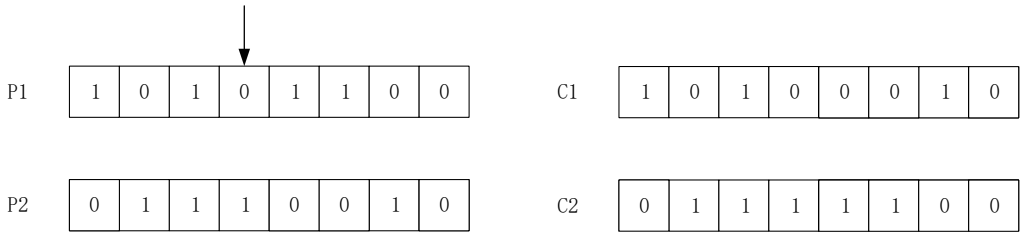

Figure 2. Genetic algorithm single point crossover.

The mutation is to change the genetic gene on the chromosome by random selection and flips the gene in the individual. If the gene is 0, the mutation is 1; if the gene is 1, the variation is 0.

2.1.5. Algorithm Termination Condition

When the operation reaches the maximum number of iterations, or the fitness of the current population reaches 0.999, the algorithm terminates.

2.2. Convolutional Neural Network

2.2.1. Convolution Layer

The convolutional layer is the core of the convolutional neural network and consists of multiple feature maps. Each feature map is composed of multiple neurons, as shown in Figure 3. Each neuron is locally connected to the feature map of the previous layer through a convolution kernel and is inner produced with the pixel at each position. The convolutional layer extracts input features through a convolution operation, the first layer extracts low-level features such as edges and lines, and higher-level convolutional layers extract higher-level features. The neurons in the first convolutional layer are locally connected to the feature map in the input layer, and the locally weighted sum is transmitted

to the nonlinear activation function, through which the output value of each neuron in the convolutional layer is calculated. The nonlinear function of the convolutional layer in the convolutional neural network usually selects the ReLU function.

$$f(x) = \max(0, x) \tag{8}$$

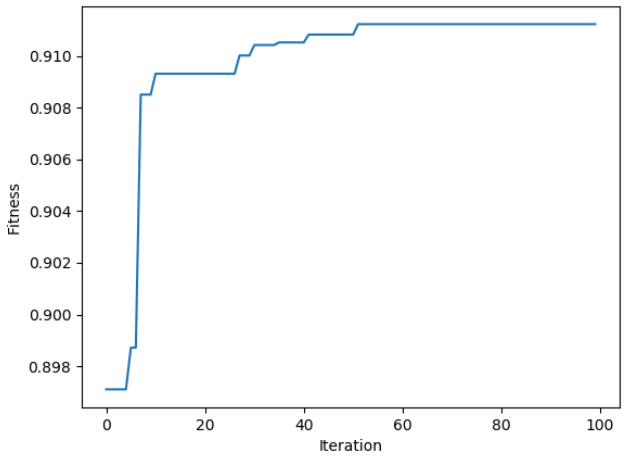

Figure 3. Optimal fitness of 100 generations.

Because the ReLU function has the characteristics of unilateral inhibition and sparse activation, in Equation (8), when $x > 0$, the output is x, the gradient is constant to 1, and there is no gradient dissipation problem. When $x < 0$, the output is 0, and the more neurons are 0, the more obvious the sparse degree of the network is and the more representative the extracted features are, which can alleviate the problem of overfitting. In the same input feature map and the same output feature map, the weights of CNN are shared, which can reduce the complexity of the model. The size of each convolutional layer in CNN (*oMapN*) satisfies the following relationship:

$$oMapN = \left(\frac{(iMapN - CWindow)}{CInterval} + 1 \right) \tag{9}$$

where *iMapN* represents the size of each input feature map, *CWindow* is the size of the convolution kernel, and *CInterval* is the sliding step size of the convolution kernel in its upper layer. In order to ensure that (9) can be divisible, the CNN network structure needs to be processed. The number of trainable parameters for each convolutional layer (*C*params) satisfies the following relationship:

$$CP\text{arams} = (iMap \times CWindow + 1) \times oMap \tag{10}$$

where *oMap* is the number of output feature maps for each convolutional layer, *iMap* is the number of input feature maps, 1 is bias, and the bias is also shared in the same output feature map. In the convolution layer, the output value of the kth neuron of the output feature map n is x_{nk}^{out} and x_{mh}^{in} represents the output value of the hth neuron of the input feature map m. Then,

$$x_{nk}^{out} = f_{\text{cov}}\left(x_{1h}^{in} \times \omega_{1(h)n(k)} + x_{1(h+1)}^{in} \times \omega_{1(h+1)n(k)} + x_{1(h+2)}^{in} \times \omega_{1(h+2)n(k)} + \ldots + b_n \right) \tag{11}$$

In Equation (11), b_n is the bias value of the output characteristic map n and $f_{\text{cov}}(\bullet)$ is the nonlinear excitation function.

In the CNN structure, the deeper the depth and the more the number of feature maps, the larger the feature space that the network can represent and the stronger the learning ability of the network, but the calculation will be more complex and prone to overfitting. Se-

lect the appropriate network depth, the number of feature maps, the size of the convolution kernel, and the step size to obtain a good model while reducing training time.

2.2.2. Pooling Layer

The pooling layer is connected after the convolution layer and consists of multiple feature maps. Each feature map is uniquely corresponding to the feature map of the previous layer without changing the number of feature maps. The convolution layer serves as the input layer of the pooling layer, and the neurons of the pooling layer are locally connected with it. The pooling layer aims to reduce the resolution of the feature map to obtain features with spatial invariance. The commonly used pooling methods are maximum pooling and mean pooling. The maximum pooling is to take the point of the maximum value in the local domain, and the mean pooling is to take the average value of all points in the local domain. The window where the pooling layer slides in the previous layer is called the pooling kernel. The size of each output feature map of each pooling layer in CNN (*DoMapN*) is

$$DoMapN = \left(\frac{oMapN}{DWindow} \right) \tag{12}$$

The size of the pooling kernel is *DWindow*. The pooling layer reduces the number of connections between convolution layers by reducing the number of neurons, which reduces the calculation amount of the network model and avoids the phenomenon of overfitting.

2.2.3. Fully Connected Layer

In CNN, after the structure of several convolutional layers and pooling layers, one or more fully connected layers are connected. Each neuron in the fully connected layer is fully connected to all neurons in the previous layer. The output can be expressed by Formula (13).

$$f(x) = W \times x + b \tag{13}$$

In Formula (13), x is the input of the fully connected layer, W is the weighting factor, and b is the bias. The activation function of each neuron in the fully connected layer generally uses the ReLU function. The output value of the last fully connected layer is passed to the output layer, and the softmax function can be used for classification. In order to avoid training overfitting, dropout technology is usually used in the fully connected layer. Through this technology, some hidden layer nodes fail. These nodes do not participate in the propagation process of CNN, which reduces the complexity of mutual adaptation between neurons and enables neuron learning to obtain more robust features.

3. Results

The software and hardware equipment and tools used in the experiment are shown in Table 1.

Table 1. Hardware and software equipment and tools.

Equipment	Version
Hardware	Windows10, Intel® Core™ i5-8250U CPU @ 1.60 GHz
Python	3.7.0
Tensorflow	2.9.1
Sklearn	1.0
Pycharm	11.0.2

3.1. Data Set

This paper uses this model to identify malicious URLs. The URL data set needs to contain benign URLs and malicious URLs. The data used in this paper are the required URL obtained by the Python crawler. PhishTank obtains 29,873 malicious URL data. PhishTank

contains a large number of suspicious phishing websites, forming a database of phishing websites. Therefore, this paper selects these data as a malicious URL data set. Alexa obtains 29,700 benign URL data. Alexa is a traffic statistics website. The website ranked according to the current traffic has relatively high security. Therefore, this paper selects the URL on the ranking as a benign URL data set.

Table 2 is a partial table of the obtained original data set, where id is the number of each url, url represents a malicious URL or a benign URL, the flag represents an identifier of url type, the flag is p indicates that the url is a malicious url, flag is n indicates that the url is a benign url.

Table 2. Original data.

Id	Url	Flag
1	http://www.gregorty.net/r4ff/	p
2	http://www.charliedonutsstevens.com /npaf/uazh/old/np/mil/NFOAA_Auth/login/jsp	p
3	http://www.charliedonutsstevens.com /live/tmp/old/np/mil/NFOAA_Auth/login/jsp/	p
4	https://secure.lcs-card.net/login	p
5	https://sotly.me/WaI/qr	p
6	https://olx-my.xyz	p
7	https://sotly.me/WaI	p
8	https://jaccs.co.jp.bjjygd.com/	p
9	http://elongiveway.com	p
...	...	...

By analyzing the characters contained in the URL, the URL data are preprocessed by observing the type and number of characters contained in the benign URL and the malicious URL, and 34 features are selected as the initial feature vector of the URL, denoted by F1 = {f1,f2, . . . ,f34}.

3.2. Feature Extraction

3.2.1. Initialization Population and Coding

The initial population size is set to 20, and the initial population is selected by random method. All candidate objects are binary-coded to obtain a chromosome. In the data set of this paper, 34 feature items are binary coded as genes of a chromosome; that is, each individual is composed of 34 0 or 1.

3.2.2. Calculation of Fitness Function

The fitness value of each individual in the population is calculated by the fitness function. In this paper, the prediction accuracy of SVM is used as the fitness function; that is, the features that can obtain higher accuracy are selected, and the features with poor effect are discarded. Set the kernel parameter to rbf, the gamma parameter to scale, and calculate the accuracy value as the fitness value.

3.2.3. Choice Operation

Through the above operation, the fitness values of 20 individuals are obtained. The roulette method is used to select the parent individual. The probability of each individual being selected is calculated by Formula (7). The larger the probability, the easier the individual is selected. The set number of parent individuals is six; that is, select six individuals as the next generation of parents.

3.2.4. Crossover and Mutation Operation

Through the above selection operation, 12 groups of parents were selected from the parent combination for crossover operation. In this paper, a single-point crossover was

used to randomly select the crossover breakpoint, and a total of 12 new individuals were obtained after the crossover. Then, 12 new individuals were mutated. Among the 34 genes of each individual, 3 genes were randomly selected for reversion, and finally, 12 individuals of the new generation population were obtained. The first eight individuals were selected to join the offspring population by order of fitness values from large to small.

3.2.5. Algorithm Termination Operation

Repeat the selection, crossover, and mutation operations of the new generation population, and set the number of iterations to 100. The algorithm is terminated when the maximum number of iterations is reached, and the result of the last generation is the result of feature extraction. In this paper, several experimental tests were carried out, and finally, 20 features with high accuracy were obtained. Table 3 is the data set of 20 features obtained after feature extraction. When the flag is 0, it represents a benign URL, and when the flag is 1, it represents a malicious URL.

Table 3. The data set after feature extraction.

Reservechar	Otherchar	Digitcount	LongSubLetter	LongSubDigit	...	Flag
5	4	12	11	9		0
7	3	2	14	2		0
6	3	9	8	6		0
6	2	0	7	0		0
12	4	6	5	2		0
5	1	7	4	7		0
5	3	6	5	6		0
5	3	4	9	4		0
... ...	... ...	... ...	... ...	... ...	...	... ...
7	4	9	9	7		1
7	4	14	7	3		1
6	2	5	6	3		1
7	3	7	6	7		1
3	1	0	9	0		1

The experimental results show that a total of 20 features are selected as F2 = {f1, f2, f3,..., f20}, including the number of reserved characters, the number of other characters, the number of numbers, the maximum length of continuous letters, the maximum length of continuous numbers, the quantitative relationship between '?' and '=', the quantitative relationship between '=' and '&', the total length of the domain name, the domain name series, the longest string length of the domain name, the number of numbers in the domain name, whether the path contains the domain name, the path series, the ratio of the longest path to the path length, whether the file name contains more than two levels of extensions, the maximum length of the sub-path, the conversion frequency of numbers and letters in the URL, the proportion of vowel–consonant characters in the URL, the proportion of number characters in the domain name to the domain name, and whether the top-level domain name is five domain names. Figure 3 shows the optimal fitness value of each generation selected during the evolution process. Through 100 iterations, the final fitness of the new generation population reached 0.91.

The genetic algorithm can quickly reach more than 90% of the optimal solution and global search capability, and parameter setting is relatively easy. Compared with the traditional method of feature extraction, it avoids easy to fall into the local optimum and solves the defect of not finding the optimal global solution.

3.3. Convolutional Neural Network

The data set completed by feature extraction is divided into 80% training set and 20% test set, and the target value is one-hot encoded, and the F2 is reshaped as the input of the network.

The parameters to be optimized in Keras-based 1DCNN are convolution kernel size and network depth. In this experiment, a single-layer convolution layer and a pooling layer were used. The convolution kernels of each layer are equal in size, and different convolution kernel sizes are set for experiments. The number of convolution kernels is determined by experience. Figure 4 shows the model accuracy for different convolution kernel sizes when the number of channels is 32.

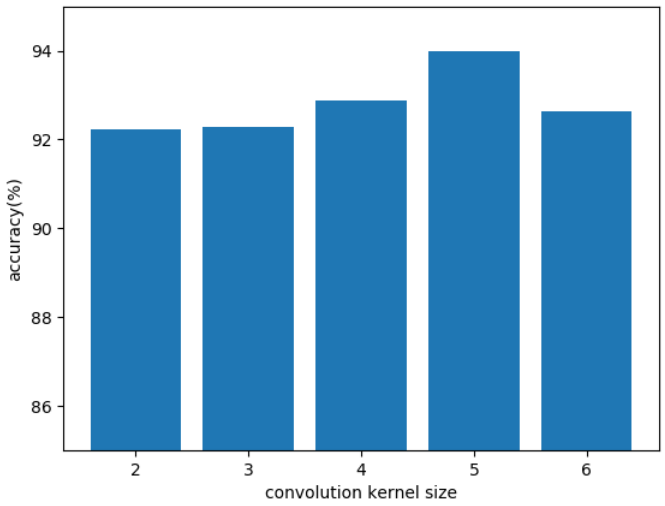

Figure 4. Model accuracy based on different convolution kernel sizes.

As can be seen from Figure 5, under the premise of an unchanged number of channels, with the increase in convolution kernel size, the accuracy of network identification first increases and then decreases. When the convolution kernel size is 5, the network has the best recognition ability, so the convolution kernel size is 5.

Conv1d_input:Input Layer	input	(None, 20, 1)
	output	(None, 20, 1)
Conv1d:Conv1D	input	(None, 20, 1)
	output	(None, 16, 64)
Maxpooling1d:MaxPooling1D	input	(None, 16, 64)
	output	(None, 8, 64)
Conv1d:Conv1D	input	(None, 8, 64)
	output	(None, 4, 64)
Maxpooling1d:MaxPooling1D	input	(None, 4, 64)
	output	(None, 2, 64)
flatten:Flatten	input	(None, 2, 64)
	output	(None, 128)
dense:Dense	input	(None, 128)
	output	(None, 2)

Figure 5. Model structure information of Model C.

In addition to the size of the convolution kernel, the depth of the network and the number of channels of the convolution kernel also affect the effectiveness of the model. Therefore, three models are established for experiments through different depths and different channel numbers, as shown in Table 3. The output of the convolution layer is obtained by the excitation function ReLU to obtain the input of the pooling layer. The process of convolution and pooling is filled in the way of 'VALID'. Set Dropout to 0.5, Adam Optimizer, learning rate to 0.0001, and epoch to 100 to train the model.

In Table 4, Model A and Model B establish a pair of convolution layer and pooling layer and two pairs of convolution layer and pooling layer, respectively. Set the same size of the convolution kernel and pooling kernel, and the step size and channel size are 32. Model C has the same number of network layers as Model B, and the channel size is set to 64 for comparison. The data set is used to verify the three models, and the model is evaluated by the prediction accuracy rate (Equation (14)) and the loss rate (Equation (15)). The accuracy rate reflects the ability of the model to judge the overall sample correctly. The categorical_crossentropy loss rate is used to train the model through the loss value of each iteration.

$$accuracy = \frac{TP + TN}{P + N} \tag{14}$$

$$loss = -\sum_{i=1}^{outputsize} y_i \cdot \log \hat{y}_i \tag{15}$$

Table 4. Different parameter settings.

Model	Convolution Kernel	Pooling Kernel	Stride	Channel Size
Model A	1×5	1×2	1	32
Model B	$1 \times 5, 1 \times 5$	$1 \times 2, 1 \times 2$	1	32, 32
Model C	$1 \times 5, 1 \times 5$	$1 \times 2, 1 \times 2$	1	64, 64

Table 5 shows the accuracy and loss rate of the three models for malicious URL recognition. It can be seen from Table 3 that when the convolution kernel size is the same, the effect is better when the network depth is 7. When the size of the convolution kernel and the depth of the network are fixed, the effect of the larger number of channels is better, so model C has a better classification effect. The model structure information of model C is shown in Figure 5.

Table 5. Test results of different network models.

Model	Model A	Model B	Model C
Accuracy	93.64	93.72	93.99
Loss	0.191	0.180	0.169

Among them, the input in the input layer is a 20-dimensional feature vector; the output of Conv1d is 16, obtained by formula (9), and the output of Maxpooling1d is 8, obtained by formula (12). After that, a pair of convolutional layers and pooling layers are calculated the same.

Figure 6 shows the loss rate obtained by using the network training iteration 100 times. It can be seen that the loss rate of the training set and the loss rate of the test set are both in a downward trend. The loss rate of the training set is 0.170, and the loss rate of the test set is 0.169, indicating that the network is moderately fitted.

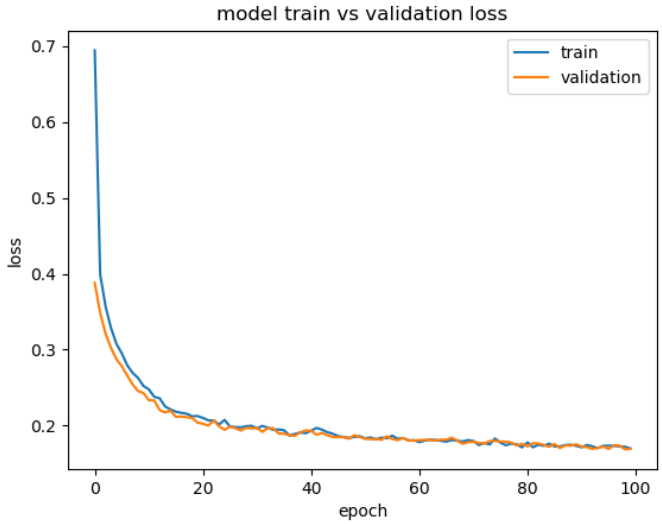

Figure 6. Loss values for training and prediction.

As shown in Figure 7, the accuracy of the prediction results is 93.99%, indicating that the model has a good effect on malicious URL classification.

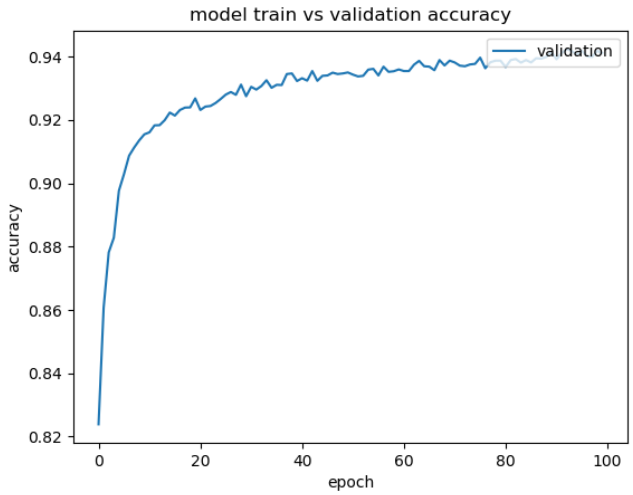

Figure 7. Accuracy of prediction results.

Table 6 shows the running results of the model (CNN (GA)) and various traditional machine learning models in this data set. The first five models are classified based on the 34 features extracted in this paper. KNN selects the optimal K value 10 through grid search, and the network structure of CNN is the same as that of CNN (GA). It can be seen from the results that the CNN model based on genetic algorithm has improved the accuracy of malicious URL recognition.

Table 6. Comparison of recognition accuracy of multiple models.

Model	KNN	GaussianNB	LogisticRegression	SVC	CNN	CNN (GA)
Accuracy	0.921	0.715	0.896	0.915	0.932	0.940

The ROC curve of the model is shown in Figure 8, obtained by

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{16}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \tag{17}$$

calculated, the AUC area of the model is 0.99, which has a predictive value.

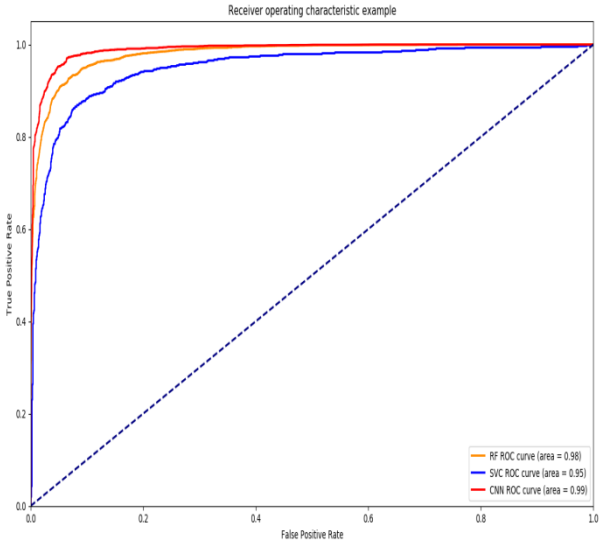

Figure 8. ROC curve.

Malicious URL classification using SVM and random forest models. The ROC curve is obtained by calculating TPR and FPR through formula (16) and formula (17). As shown in Figure 8, the AUC of SVM is 0.95, the AUC of random forest is 0.98, and the AUC of the CNN model used in this paper is 0.99. It is concluded that the prediction results of this model are valuable and better than other models.

4. Conclusions and Future Research

In order to improve the accuracy of malicious URL recognition, this paper proposes a model of a convolutional neural network based on a genetic algorithm. Firstly, this paper extracts the grammatical features, structural features, and probabilistic features in the URL text by comparing the malicious URL with the benign URL, reduces the dimension of the features by genetic algorithm to obtain the final 20 vectors, and then uses the convolutional neural network to classify and identify the 20 features. Finally, compared with the traditional machine learning algorithm, the recognition rate of malicious URL detection is improved. The experimental results show that the accuracy of the model in malicious URL classification is 93.99%, which achieves the expected classification effect.

In malicious URL detection, the model proposed in this paper has a better effect than directly detecting the entire URL text content by extracting insignificant features in the URL. Second, the genetic algorithm is used to reduce the dimension of the extracted features, and the redundant features are removed to reduce the computational overhead. Third, when using the advantages of convolutional neural networks, the classification effect is better while sharing parameters in the convolutional layer, reducing the use of parameters.

However, there are still some limitations in the experiment. The genetic algorithm can obtain the global optimal feature subset and effectively improve the accuracy of malicious URL recognition. However, there are many parameters used in the calculation process; the calculation amount is large, and the experiment takes a long time. This model cannot detect targets in real time. Therefore, in order to solve these problems, our further research work is to optimize the model around how to reduce the feature vector space while reducing the time complexity and exploring the method of detecting malicious URLs in real time.

Author Contributions: Conceptualization, T.W. and Y.X.; methodology, T.W. and Y.X.; software, M.W. and Z.Z.; validation, T.W. and Y.X.; formal analysis, T.W. and Y.X.; investigation, T.W. and

Y.X.; resources, T.W. and Y.X.; data curation, Y.X.; writing—original draft preparation, Y.X.; writing—review and editing, T.W.; visualization, Y.X.; supervision, T.W.; project administration, T.W.; funding acquisition, T.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Subasi, A.; Balfaqih, M.; Balfagih, Z.; Alfawwaz, K. A comparative evaluation of ensemble classifiers for malicious webpage detection. *Procedia Comput. Sci.* **2021**, *194*, 272–279. [CrossRef]
2. Sayamber, A.B.; Dixit, A.M. Malicious URL detection and identification. *Int. J. Comput. Appl.* **2014**, *99*, 17–23.
3. Jian, L.; Gang, Z.; Yunpeng, Z. Design and implementation of malicious URL multi-layer filtering detection model. *Inf. Netw. Secur.* **2016**, *1*, 6.
4. Vundavalli, V.; Barsha, F.; Masum, M.; Shahriar, H.; Haddad, H. Malicious URL detection using supervised machine learning techniques. In Proceedings of the 13th International Conference on Security of Information and Networks, Merkez, Turkey, 4–7 November 2020; pp. 1–6.
5. Rahman SS, M.M.; Islam, T.; Jabiullah, M.I. PhishStack: Evaluation of stacked generalization in phishing URLs detection. *Procedia Comput. Sci.* **2020**, *167*, 2410–2418. [CrossRef]
6. Zeyu, L.; Yong, S.; Zhi, X. Malicious URL recognition based on machine learning. *Commun. Technol.* **2020**, *53*, 5. (In Chinese)
7. Pham TT, T.; Hoang, V.N.; Ha, T.N. Exploring efficiency of character-level convolution neuron network and long short term memory on malicious URL detection. In Proceedings of the 2018 VII International Conference on Network, Communication and Computing, Taipei City, Taiwan, 14–16 December 2018; pp. 82–86.
8. Chen, Z.; Liu, Y.; Chen, C.; Lu, M.; Zhang, X. Malicious URL detection based on improved multilayer recurrent convolutional neural network model. *Secur. Commun. Netw.* **2021**, *2021*, 9994127. [CrossRef]
9. Li, T.; Kou, G.; Peng, Y. Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Inf. Syst.* **2020**, *91*, 101494. [CrossRef]
10. Kumi, S.; Lim, C.H.; Lee, S.G. Malicious URL detection based on associative classification. *Entropy* **2021**, *23*, 182. [CrossRef]
11. Raja, A.S.; Vinodini, R.; Kavitha, A. Lexical features based malicious URL detection using machine learning techniques. *Mater. Today: Proc.* **2021**, *47 Pt 1*, 163–166.
12. Joshi, A.; Lloyd, L.; Westin, P.; Seethapathy, S. Using lexical features for malicious URL detection—A machine learning approach. *arXiv* **2019**, arXiv:1910.06277.
13. Kang, C.; Huazheng, F.; Yong, X. Malicious URL identification based on deep learning. *Comput. Syst. Appl.* **2018**, *27*, 27–33.
14. Yuan, J.T.; Liu, Y.P.; Yu, L. A novel approach for malicious URL detection based on the joint model. *Secur. Commun. Netw.* **2021**, *2021*, 4917016. [CrossRef]
15. Le, H.; Pham, Q.; Sahoo, D.; Hoi, S.C. URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv* **2018**, arXiv:1802.03162.
16. Yuan, J.; Chen, G.; Tian, S.; Pei, X. Malicious URL detection based on a parallel neural joint model. *IEEE Access* **2021**, *9*, 9464–9472. [CrossRef]
17. Zhao, G.; Wang, P.; Wang, X.; Jin, W.; Wu, X. Two-dimensional code malicious URL detection method based on decision tree. *Inf. Secur. Technol.* **2014**, *5*, 36–39. (In Chinese)
18. Liu, C.; Wang, L.; Lang, B.; Zhou, Y. Finding effective classifier for malicious URL detection. In Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences, Wuhan, China, 13–15 January 2018; pp. 240–244.
19. Lin, H.L.; Li, Y.; Wang, W.P.; Yue, Y.L.; Lin, Z. Efficient malicious URL detection method based on segment pattern. *Commun. J.* **2015**, *36*, 141–148. (In Chinese)
20. Gabriel, A.D.; Gavrilut, D.T.; Alexandru, B.I.; Stefan, P.A. Detecting malicious URLs: A semi-supervised machine learning system approach. In Proceedings of the 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 24–27 September 2016; IEEE: New York, NY, USA, 2016; pp. 233–239.