Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets☆

Xi Xiao[a,b], Wentao Xiao[a,b], Dianyan Zhang[a,b], Bin Zhang[b], Guangwu Hu[c,*], Qing Li[b,d], Shutao Xia[a,b]

[a] Shenzhen International Graduate School, Tsinghua University, China
[b] Peng Cheng Laboratory, Shenzhen, China
[c] School of Computer Science, Shenzhen Institute of Information Technology, Shenzhen, China
[d] Southern University of Science and Technology, Shenzhen, China

ARTICLE INFO

Article history:
Received 4 December 2020
Revised 12 May 2021
Accepted 9 June 2021
Available online 16 June 2021

Keywords:
Phishing websites detection
Imbalanced dataset
Generative adversarial network
Convolutional neural network
Multi-head self-attention

ABSTRACT

Phishing websites belong to a social engineering attack where perpetrators fake legitimate websites to lure people to access so as to illegally acquire user's identity, password, privacy and even properties. This attack imposes a great threat to people and becomes more and more severe. In order to identify phishing websites, many proposals have shown their merits. For example, the classical proposal CNN-LSTM received a very high precision by combining Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) together. However, despite CNN achieved great success in AI area, LSTM still exists the biases issue since it always treats the later features much more important than the former ones. In the meanwhile, as the self-attention mechanism can discover the text's inner dependency relationships, it has been widely applied to various tasks of deep learning-based Natural Language Processing (NLP). If we treat a URL as a text string, this mechanism can learn comprehensive URL representations. In order to improve the accuracy for phishing websites detection further, in this paper, we propose a novel Convolutional Neural Network (CNN) with self-attention named self-attention CNN for phishing Uniform Resource Locators (URLs) identification. Specifically, self-attention CNN first leverages Generative Adversarial Network (GAN) to generate phishing URLs so as to balance the datasets of legitimate and phishing URLs. Then it utilizes CNN and multi-head self-attention to construct our new classifier which is comprised of four blocks, namely the input block, the attention block, the feature block and the output block. Finally, the trained classifier can give a high-accuracy result for an unknown website URL. Overall thorough experiments indicate that self-attention CNN achieves 95.6% accuracy, which outperforms CNN-LSTM, single CNN and single LSTM by 1.4%, 4.6% and 2.1% respectively.

☆ This work is supported in part by the National Key Research and Development Program of China (2018YFB1800204, 2018YFB1800601), the National Natural Science Foundation of China (61972219, 61771273), Natural Science Foundation of Guangdong Province (2021A1515012640), and the R&D Program of Shenzhen (JCYJ20190813174403598, SGDX20190918101201696, JCYJ20190813165003837).

* Corresponding author.:
E-mail address: hugw@sziit.edu.cn (G. Hu).

https://doi.org/10.1016/j.cose.2021.102372

1. Introduction

The phishing website is an online social engineering attack leading to privacy leakage, identity theft and property damage by pretending to be a legitimate entity (Peng et al., 2019; Verma and Das, 2017). Phishers aim to trick online users so as to catch their financial information such as credit card numbers, password, etc. Rajab (2018), which impose a great threat to Internet users, and this phenomenon is becoming more and more serious now. According to the Phishing Activity Trends Report (Report from APWG, 2018) which was published by Anti-Phishing Working Group (APWG), the total numbers of phishing sites detected by APWG in Q1, Q2, Q3, Q4 of 2018 reach to 263538, 233040, 151014 and 138328, separately. Meanwhile, the RSA Online Fraud Report (Rsa online fraud report, 2019d) reveals that the phishing attacks have cost global organizations $4.6 billion in losses in 2015, and this number has been increasing in recent years.

To detect phishing websites, industry and academic communities have made their great efforts. For example, Google has set up a blacklist which gathers a large number of reported phishing websites for phishing detection and applied the list in its own browser Chrome (Liang et al., 2016). Other companies take other resorts such as toolbars or browser extensions to identify and block phishing websites (Cui et al., 2017). Besides, corporations such as Panda (Panda security, 2019) and McAfee (Mcafee phishing protection, 2019) have integrated anti-phishing service into their anti-virus software.

In the meantime, many researches also have proposed various methods from different academic angles. For instance, the simplest practice, blacklist or whitelist, takes effect by setting up an illegal or legitimate Uniform Resource Locators (URLs) list. But the weakness of this approach is that the lists cannot cover all phishing websites and this practice cannot defy the tricks such as the zero-day attack (Aravindhan et al., 2016; Mohammad et al., 2012). In order to overcome this drawback, many machine learning methods, e.g., Naive Bayer, J48 tree, Random Forest, Logistic Regression, Support Vector Machine, AdaBoostM1 Zhang et al. (2011) and etc., have shown great advantage by extracting features from URLs or webpage contents and training the classifier to give a final verdict. In the extracting and training procedures, they basically rely on manually prepared expert knowledge, which may result the final verdict very subjective. As the improvement, deep learning methods, such as Long Short Term Memory (LSTM), Deep Belief Networks and Convolutional Neural Network (CNN) have been applied in phishing detection to avoid the subjectivity caused by the manually extracted features (Correa Bahnsen et al., 2017; Peng et al., 2019; Zhang and Li, 2017). The underlying principle in deep learning for this improvement is that the features are not designed by human engineers, but learned from the data by a general-purpose learning procedure (Lecun et al., 2015). However, these deep learning approaches still face the problem of unsatisfied accuracy.

Moreover, most of the machine learning and deep learning methods mentioned above do not consider the problem of imbalanced training datasets. The problem results from the fact that legitimate URLs are greatly more than the phishing ones. In this situation, the classifier learns more features from the majority class which may cause the biased results (Verma and Das, 2017).

In order to balance the training datasets and improve the accuracy of phishing websites identification, in this paper, we propose self-attention CNN, a high-accuracy phishing websites detection approach via CNN (Ketkar, 2017) and Multi-Head Self-Attention (Vaswani et al., 2017). self-attention CNN first takes Generative Adversarial Network (GAN) to produce phishing URLs so as to balance the datasets between phishing and legitimate URLs (Chawla et al., 2002; Goodfellow et al., 2014). Next, we combine the deep learning network of CNN and multi-head self-attention together to build our classifier. There are four important blocks, i.e., the input block, the attention block, the feature block and the output block, in the classifier. On the balanced datasets, the input block transforms URLs into matrixes, which are duplicated. Subsequently, the two duplicated copies are respectively fed into the attention block and the feature block to get attention weights and learn features. At last, the output block gives the detection result. In the specific, we adopt the multi-head self-attention mechanism in the attention block, which can find the inner dependency relationships between different characters of URLs. This helps our method learn comprehensive URL representations. The extensive experiments show that our generated data can balance the dataset and our method can detect phishing websites more precisely. Accuracy of self-attention CNN achieves 95.6% which is higher than those of CNN-LSTM, single CNN and single LSTM by 1.4%, 4.6% and 2.1% respectively.

The main contributions of our work are as follows:

- We use GAN to generate synthetic URLs to make the training dataset balanced. These URLs are so similar with real-world phishing URLs that they greatly facilitate training an unbiased classifier.
- CNN and multi-head self-attention are combined to construct one new classifier, which can improve the results. To the best of our knowledge, this is the first attempt in phishing websites detection.
- We conduct a series of experiments, which illustrate that our approach obtains high accuracy in phishing URLs identification. The result also proves that our method is superior than the classic schemes.

To evaluate our proposed self-attention CNN model, we pose five questions to discuss the performance of our method:

Q1: Do different ratios between real-world legitimate URLs and real-world phishing URLs impact the classification results?

Q2: How about the experiments on the dataset including phishing URLs generated by GAN and real-world URLs and on the dataset with only real-world URLs?

Q3: How do parameters influence the performance of our classifier?

Q4: What's the situation when our classifier is compared with the other existing schemes?

Q5: Are generative URLs created by GAN more useful for classification than those made by the other methods?

The rest of paper is organized as follows: Section 2 summarizes different methods for detecting phishing websites. Next Section 3 introduces some background, e.g. imbalanced data classification, Convolutional Neural Network and multi-head self-attention. In the following, Section 4 describes our method in detail including using GAN to balance the dataset and constructing our new network. Further, Section 5 shows our experiments and results on different datasets and makes some comparisons. Finally, Section 6 concludes the whole paper and points out our future work.

2. Related work

Nowadays, many efforts have shown their merits from different views, which mainly can be divided into four categories.

Blacklist-based methods are widely used by many companies and browsers. They record a lot of phishing websites via different techniques, such as searching known phishing characteristics in the web and etc. Correa Bahnsen et al. (2017), Zhang et al. (2008), Ma et al. (2009). For example, Google Safe Browsing holds its own blacklist to block recorded phishing websites when users access to them. PhishTank (Ludl et al., 2007) is an online open blacklist in which users can upload phishing websites they meet and download the phishing websites recorded. One drawback of such approach is that only recorded phishing websites can be blocked and the zero-ady attack can evade. However, most phishing websites can be recorded after 10 h. During that time they can lure many victims and do damage to people (Cui et al., 2017).

Graph-based methods aim to detect phishing websites by finding targets with graphs. These methods always build a web graph in which the node represents a webpage, and the edge between nodes refers to the hyperlink from one page to another (Wenyin et al., 2012). Then they create a relationship matrix according to the graph (Futai et al., 2016). The row sum and column sum of this matrix are the out-degree and the in-degree respectively (Gowtham et al., 2017). With these degrees, they calculate some metrics to find the target website. Finally, if the website is different with the target, it is the phishing one. However, the built graphs increase cost of manpower and material resources. For example, Wenyin et al. (2012) collected webpages with either direct or indirect association with a given suspicious webpage. With these webpages, they created the "parasitic" community and then found the phishing target, that is, the page with the strongest parasitic relationship. Besides, the work (Futai et al., 2016) defined the AD-URL graph and utilized Markov random field to find the phishing target.

Rule-based methods always extract features manually to construct feature matrixes (Zuhair and Selamat, 2019). Then they use some machine learning algorithms to do a binary classification with matrixes as inputs, i.e., classify websites into either legitimate or phishing websites (Abutair and Belghith, 2017; Dunham, 2009). These methods have two main aspects, features and algorithms. The features include URL, webpage content, and the third-part information features. URL features generally contain the length of full URL, the domain name, the directory, the file and the number of symbols (Le et al., 2010; Tan et al., 2016). The work (Marchal et al., 2016) focuses on the construction of URL and from which extracts protocol, fully qualified domain name, registered domain name, main level domain, etc.. Webpage content features are from HTML code and HTTP cookies, etc. Corona et al. (2017). They comprise page source code, images, textual content, text formatting, HTML tags, CSS, and website logos, etc. Jain and Gupta (2017). Moreover, the third-part information refers to IP, connection speed, or registrar information which can be received from service providers (Blum et al., 2010). With respect to the algorithms, Sequential Minimum Optimization (SMO), J48 tree, Random Forest (RF), Logistic Regression (LR), Multilayer Perceptron (MLP), Naïve Bayes (NB), Support Vector Machine (SVM) and AdaBoostM1, etc., are utilized for phishing detection (Srinivasa Rao and Roshan Pais, 2018). These methods can block zero-day attack. In detail, CANTINA was developed as an ensemble classifier including NB, SVM and LR, etc. Zhang et al. (2007). It extracted around 15 features and got 92% True Positive Rate and 1.4% False Positive Rate. Gowtham and Krishnamurthi (2014) leveraged 17 features and developed the SVM classifier. It achieved 99.6% True Positive Rate and 0.44% False Positive Rate. Further, Marchal et al. (2014) used SVM, RF, C4.5 and RIPPER (JRip) algorithms with 12 features. Zhang et al. (2014) applied SMO, LR, NB and RF with language independent features.

Neural network-based methods usually regard phishing URLs as sentences. They transform URLs into input matrixes via an embedding layer (Correa Bahnsen et al., 2017). Then different neural networks can be employed to calculate the classification results. Especially, Correa Bahnsen et al. (2017) treated URLs as sentences and intercepted 150 characters to make up input matrixes. Next the matrixes were fed into LSTM for classification and the accuracy of this method was better than RF. Peng et al. (2019) designed a network with CNN and LSTM, called CNN-LSTM. CNN-LSTM used CNN to extract local correlation features and adopted LSTM to learn the sequential dependency from character sequences. It combined some features and the classification result of deep learning into multidimensional features, and input them to XGBoost to get the classification result. Experiments show CNN-LSTM gets better results than single CNN, CNN-CNN, single RNN, RNN-RNN, single LSTM, LSTM-LSTM and CNN-RNN. From the experiments, they found that CNN-LSTM achieved the highest accuracy. In addition, Chen et al. (2018) did phishing detection research based on PSO-BP (Particle Swarm Optimization-Back Propagation) Neural Network and received high accuracy. Wei et al. (2020) proposed a convolutional neural network (CNN) to detect malicious URL. Their model input only the URL text. Thus it is very fast and can be used on mobile devices. Su (2020) leverages Long Short-Term Memory (LSTM) to detect phishing websites. Compared with others methods, neural network-based methods learn features from data automatically, other than extract features by human engineers (Lecun et al., 2015).

3. Background

In this section, we introduce the imbalanced data classification problem and list some approaches to solve the problem. Next, we illustrate the construction of CNN. Finally, multi-head self-attention is explained.

3.1. Imbalanced data classification

Real-world datasets are predominately composed of ænormalg examples with only a small percentage of æabnormalg examples (Chawla et al., 2002). Hence, the dataset for classification often consists of different categories with unequal quantity, i.e., the dataset is imbalanced. When we use an imbalanced dataset for classification, the classifier trends to classify all test data into the majority class which leads to biased results. Recently, many ideas have been put forward to deal with the imbalanced data classification problem.

In the aspect of the dataset, re-sampling and re-partition are two widely used methods. SMOTE (Chawla et al., 2002) is a simple and effective method for re-sampling. It takes a random point on the line between the samples of the minority class and their adjacent samples to generate new samples of the minority class. However, generating new samples takes a lot of time and may result in overfitting (the classifier learns more information from the minority class and classifies most test data into the minority class). Thus new ideas are brought forward. Chen et al. (2005) discarded some support vectors of the majority class in order to make the numbers of support vectors balanced and then improve the recognition rate of the minority class. This method might remove some important information for classification when support vectors are removed. Chan and Stolfo (1998) repartitioned the training set into some balanced sub-classes and trained sub-classifiers respectively. Then he used meta learning to integrate the results from the sub-classifiers. There have been some algorithms represented to deal with imbalance data classification. One-class learning (Schölkopf et al., 2001) is an example which learns the production model of the minority class rather than a discrimination model.

Generative Adversarial Network (GAN) is a kind of deep learning model to produce data for the imbalanced data classification problem. It is composed of a generator and a discriminator (Goodfellow et al., 2014). The generator creates some samples and wants the discriminator to regard the samples as real-world. Relatively, the discriminator aims to distinguish generative samples from real-world data. Figure 1 shows the construction of GAN in which G represents the generator, D refers to the discriminator, $G(z)$ is the output of the generator with input z, and $D(x)$ is the output of the discriminator with input x. G and D are constructed with some neural networks. Firstly, some real-world samples, X, are put into the discriminator, D, in order to make $D(X)$ near 1. At the same time, the Generator, G, receives some input noise, z, as input and gives $G(z)$ as output which is some generative samples. Then we use $G(z)$ to train D. D tries to make the results of $G(z)$ near 0 and those of X near 1. However, G tries to make D classify $G(z)$ as 1. Finally, if D cannot classify $G(z)$ and X exactly, the training process is finished. The trained G can be used for generating datasets. Recently, GAN have been used to generate pictures and do data enhancement.

3.2. Convolutional neural network

Convolutional Neural Network (CNN) is a feedforward neural network. In it, there are two kinds of layers, i.e., convolutional layers and pooling layers. Convolutional layers learn features with kernels which can respond to not only a pixel but also its neighbors. Pooling layers always include the max-pooling and the average pooling which can screen for features.

CNN always receives a $h \times h \times c$ image, where h is the height and width of the image and c is the number of channels. CNN has n kernels with size $l \times m$, where l is smaller than h, and m can not be larger than c. Kernels are convolved with the image to produce n feature maps of size $h - l + 1$. Further, the max-pooling chooses the max feature map from n feature maps or the average pooling calculates the mean of n feature maps.

In 2014, CNN was introduced in sentence classification (Kim, 2014). Firstly, sentences were transformed into 2-dimensional matrixes via embedding. After that, the matrixes were put into a convolutional layer and then into a max-pooling layer. Finally, a fully connected layer helped to get the classification result. Figure 2 demonstrates an example of CNN doing sentence classification.

3.3. Multi-head self-attention

The attention mechanism comes from the study of human vision. Due to the bottleneck of information processing, humans always selectively focus on some parts of the whole information while ignore other visible parts. This mechanism is often referred to as the attention mechanism. In recent years, the attention mechanism has been widely applied to various tasks of deep learning-based Natural Language Processing (NLP).

We can use the key-value pair format to represent the input information, where the "key" is used to calculate the attention distribution and the "value" is used to generate the selected information. There are some variations of the attention mechanism. Multi-head attention uses multiple queries to compute the selection of multiple information from the input in parallel. Each attention weight focuses on a different part of the input. Self-attention makes the key equal to the value. It pays more attention to the inner structure of a query. Recently, Google combined these two attention mechanisms, which we call multi-head self-attention (Vaswani et al., 2017). Multi-head self-attention is the attention mechanism relating different positions of a single sequence in order to compute the representation of the sequence. Figure 3 gives an example of multi-head attention. We can use equations as follows to calculate it:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{1}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2}$$

$$Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{3}$$

$$softmax(X)_{ij} = \frac{\exp(X_{ij})}{\sum_j \exp(X_{ij})} \tag{4}$$

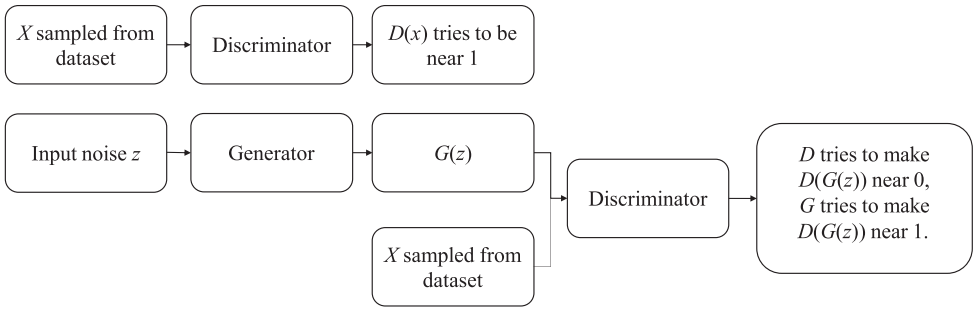


Fig. 1 – The construction of GAN.

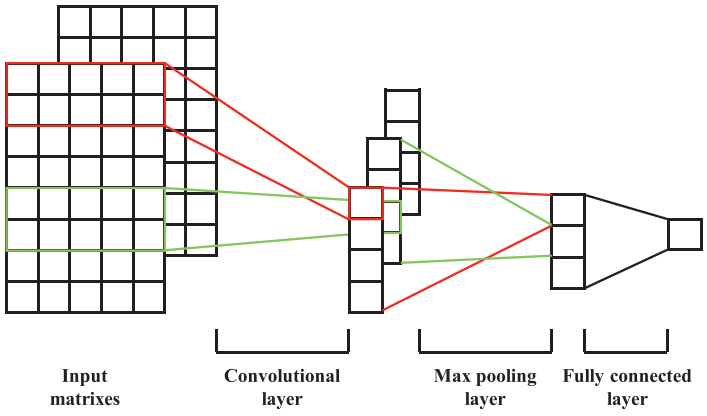


Fig. 2 – Using CNN to do sentence classification.

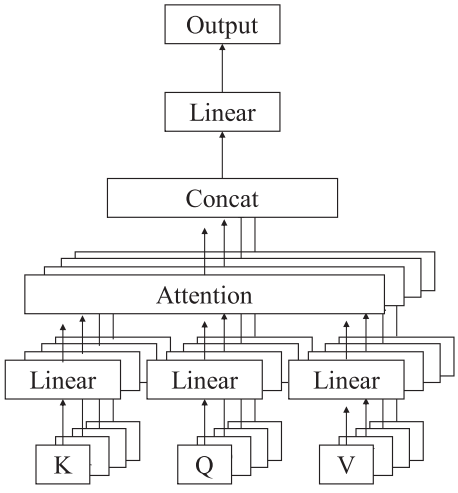


Fig. 3 – The construction of multi-head attention.

where W_i^Q, W_i^K, W_i^V, W^O are parameter matrixes, d_k is the dimension of V. In the above equation, if Q, K, V are the same, the mechanism is multi-head self-attention, otherwise it is only called as multi-head attention. Compared with LSTM, the multi-head self-attention mechanism has less calculation, so that it can be trained faster than LSTM. Furthermore, LSTM treats the later features in a query much more important than the former features, but the multi-head self-attention mechanism calculates weights to express the different importance for each feature, which is believed to be more suitable in phishing websites detection scenario.

4. Methodology

In order to reach a high-accuracy result for phishing URL detection, we make improvement in the training dataset and the construction of the classifier. Briefly, we firstly use GAN to generate real phishing URLs and form the balanced training dataset along with real-world normal URLs. In the meanwhile, based on the balanced dataset, we design a CNN and multi-head self-attention combined classifier which can take URLs as input and give a verdict (positive/negative) as output. The detailed description of the CNN and multi-head self-attention combined classifier is shown in Section 4.2. There are four important blocks, i.e., the input block, the attention block, the feature block and the output block in the classifier. Figure 4 shows the construction of our deep network, where N and M are the numbers of attention blocks and feature blocks in our method.

4.1. *Balancing training datasets with GAN*

As we know, a URL or a website address is formed with 84 kinds of characters (a-z, A-Z, 0-9, -. !*'();:&=+$,/?#[]) (Lin et al., 2017). Hence, we can leverage one-hot encoding (Money Harris and Sarah, 2007) to transform a character in the URL into a 84-dimension vector. For example, ăag is represented by the

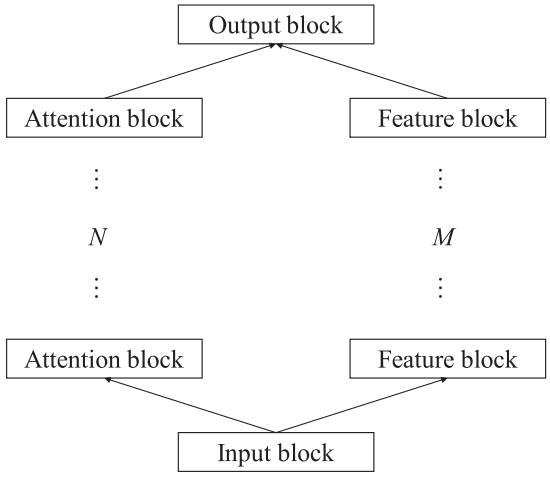


Fig. 4 – The construction of our deep network.

	h	t	t	p	…	c	o	m
⋮	0	0	0	0		0	0	0
3	0	0	0	0		1	0	0
⋮	0	0	0	0		0	0	0
8	1	0	0	0		0	0	0
⋮	0	0	0	0		0	0	0
13	0	0	0	0		0	0	1
⋮	0	0	0	0		0	0	0
15	0	0	0	0		0	1	0
16	0	0	0	1		0	0	0
⋮	0	0	0	0		0	0	0
20	0	1	1	0		0	0	0
⋮	0	0	0	0		0	0	0

Fig. 5 – An example of one-hot encoding.

vector whose first number is 1 and the other 83 numbers are 0. Then we list the column vectors according to the order of the characters in the URL to get the input matrix. Figure 5 presents an example of one-hot encoding.

With the input matrixes, we train GAN (Goodfellow et al., 2014) and take the trained generator for phishing URLs generation. Let G represent the generator of GAN, and D represent the discriminator of GAN. Both G and D are made up with some fully connected neural networks. The training process is described below:

- Sample m noise, Z from Gaussian distribution randomly. Use z_i to represent one of Z, where i is from 1 to m.
- Put Z into G and get the output $G(Z)$. $G(Z)$ are some 84-dimension vectors. For each vector, the maximum of the 84 numbers is changed to 1 and the others are changed to 0. Hence, one vector can represent a URL.
- Sample m real-world phishing URL vectors, X. Let x_i represent one of X, in which i is from 1 to m.
- Use X and $G(Z)$ to train D. X belongs to the real-world class and $G(Z)$ are a part of the generative class. Therefore, D is doing binary classification. Update D by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + \log(1 - D(G(z_i)))] \tag{5}$$

- Sample some noise, Z from Gaussian distribution randomly and put them into G to get the output $G(Z)$. Update G by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z_i)))] \tag{6}$$

- Train D and G alternately for some times. If the accuracy of D maintains 0.5, the training can be finished. It is meant that D cannot distinguish the difference of real-world phishing URLs and generative phishing URLs, so G can be used to generate phishing URLs.

After the training procedure of GAN is finished, we put noise randomly producted from Gaussian distribution into G to generate phishing URLs to train the classifier of phishing websites detection.

Both generator and discriminator are multilayer perceptrons (MLP). The input of generator is a batch of random noise vectors. The dimension of each noise vector is 100, and each element in the vector is between 0 and 1. The output of generator is a batch of the 84-dimension encoding of generative phishing URLs. Each generative phishing contains 99 characters and each character is encoded as 84-dimension vector. The generator is a 3-layer MLP. In the first and second fully-connected layer, the dimension of output feature is 128, the activation is LeakyReLU with alpha equals to 0.5, and the end of the layer is the batch normalization with momentum equals to 0.5. In the third layer, the activation is tanh, the output dimension is 8316 (99 times 84). As for the discriminator, it is a 2-layer MLP. In the first layer, the input dimension and output dimension are 8316 and 128, and the activation is LeakyReLU with alpha equals to 0.5. In the second layer, the output dimension is 1, and the activation is sigmoid.

In addition, GAN is independent as a part of data sample enhancement so as to solve data imbalance problems. Thus, GAN can combine with any existing classifiers to improve classification accuracy. In our proposal, we designed a novel neural network architecture which combines self-attention mechanism from natural language processing field and Convolution Neural Network (CNN) from the deep learning field. Moreover, the Self-attention and CNN can learn sample feature weights and sample features simultaneously, which results our classifier can integrate the results of both and output more accurate result than existing methods.

One of the limitations of GAN is that GAN requires additional time. However, GAN is only used to generate phishing URLs and balance the datasets between phishing and legitimate URLs. In the training phase, we use the balanced datasets to train self-attention CNN. In the detection phase, the real unknown URLs feed into self-attention CNN to get the labels. No generated data are used, that is GAN is not utilized in the detection phase. Nevertheless, GAN brings more time in the training phase, but does not cause additional overhead during phishing detection.

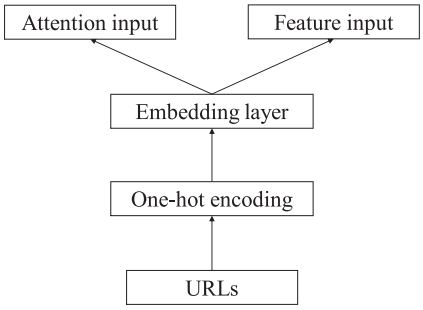


Fig. 6 – The construction of the input block.

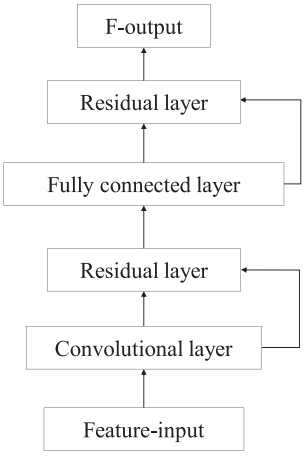


Fig. 7 – The construction of the feature block.

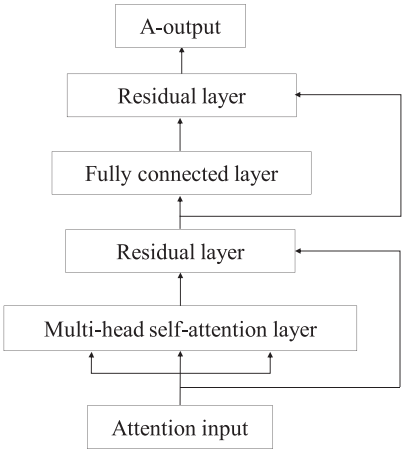


Fig. 8 – The construction of the attention block.

4.2. *Building classifier with CNN and multi-head self-attention*

In order to improve the accuracy of the final judgment, we combine CNN and multi-head self-attention to build our classifier, the construction of which is presented in Fig. 4. Generally, this classifier is composed of four blocks, the input block, the attention block, the feature block and the output block. The input block transforms URLs into matrixes. The output matrixes of this block are duplicated into two copies and then put into the attention block and the feature block to calculate attention weights and learn features at the same time. Finally, the attention weights and the features are fed into the output block to achieve final results. In these blocks, we use five kinds of network layers, the embedding layer, the convolution layer, the multi-head self-attention layer, the residual layer and the fully connected layer. Importantly, we adopt the multi-head self-attention mechanism in the attention block, which can find the inner dependency relationships between different characters of URLs. This helps our method learn comprehensive URL representations.

Firstly, in the input block which is shown as Fig. 6, URLs are transformed into initial matrixes with one-hot encoding described before. The structure of GAN consists of generator and discriminator. Then the matrixes are put into an embedding layer which is constructed with some fully connected layers to decrease the dimension of these matrixes. The output of the input block is a batch of matrices with shape (99, 84). Each matrix is the embedding of a sample (i.e., URL). Each output matrix from the embedding layer is the representation of one URL and is duplicated into two copies. The two copies are named as attention input and feature input because they become the inputs of the attention block and the feature block respectively.

Secondly, the feature input is then put into the feature block. Figure 7 explains the feature block which can learn features from input URLs. The feature block comprises a convolution layer, a residual layer, a fully connected layer and a residual layer. The convolution layer used in the feature block is 1D convolution with æsameg padding, and the size of the kernel is 3. There are 5 kernels in convolution layer with which our method can learn features from different feature space. And then we use the max-pooling to choose the feature from the 5 feature maps. Subsequently a residual layer (He et al., 2016) is supplied after the convolution layer which adds the input of the convolution layer with the output of the convolution layer. The function of the residual layer is to solve the degeneration problem: with the network depth increasing, the accuracy gets saturated (which might be unsurprising) and then degrades rapidly. After that, the output is put into a fully connected layer to do linear transformation :

$$y = max(0, w^T x + b) \tag{7}$$

Similarly, there is a residual layer after the fully connected layer. We call the final output F-output.

At the same time, the attention input is fed into the attention block. The attention block is composed of a multi-head self-attention layer, a residual layer, a fully connected layer and a residual layer. Fig. 8 gives the construction of the attention block in detail. As to the attention block, all of the hidden dimension and the output dimension of fully-connected layer is 64. The multi-head self-attention layer uses the attention input to calculate feature weights. In this work, Q, K, V of the multi-head attention layer are the same which are all attention-input and we set the number of heads as 8. Thus this multi-head attention is the multi-head self-attention. In this block, the multi-head self-attention layer can learn the inner

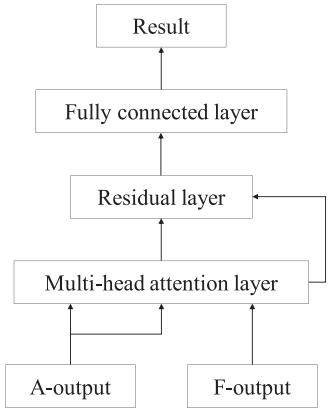


Fig. 9 – The construction of the output block.

structure of URLs. After that, one residual layer is set to solve the degeneration problem. Then the fully connected layer is leveraged to do a linear transformation. Finally, another residual layer gives the outpu of this block. The first residual layer is the sum of the input of the attention block and the output of the multi-head self-attention layer. The second residual layer is the sum of the output of the fully connected layer and the output of the first residual layer. This output is called A-output, which demonstrates the different importance of each part of URLs.

Finally, we put A-output of the attention block and F-output of the feature block into the output block. The construction of the output block is shown in Fig. 9. A-output is the output of the attention block, and F-output denotes the output of the feature block (i.e., CNN). The output block is composed of a multi-head self-attention layer, a residual layer, and a fully connected layer. In the fully connected layer, the output dimension is 64, and the activation is ReLU. As for the multi-head self-attention layer, K and V are both A-output, V is F-output, the hidden dimension is 64, and the number of heads is 8. The residual layer is the sum of the output of multi-head self-attention layer and F-output.

$$y = \frac{1}{1 + e^{-(w^T x + b)}} \tag{8}$$

The output of the Sigmoid function is a number between 0 and 1. If the number is greater than 0.5, the input URL is classified as a legitimate URL; otherwise if the number is less than 0.5, the input URL is identified as a phishing URL.

Overall, the self-attention CNN classifier consists of the input block, M feature blocks (CNN), N attention blocks, and the output block. CNN and the multi-head self-attention is combined in the output block.

5. Experiments

In this section, we first introduce datasets and metrics in our experiments. Then, we do some experiments to testify if GAN can be used to generate URLs for balancing datasets and training classifier more precisely. Finally, we compare some deep learning networks with our new network on different datasets, which proves our classifier is more accurate.

Table 1 – The composition of training dataset.

Dataset	Real-world phishing URLs	Generative phishing URLs	Real-world legitimate URLs	Ratio
D_{01}	8003	56027	64030	8
D_{02}	7114	56916	64030	9
D_{03}	6403	57627	64030	10
D_{04}	704	63326	64030	90
D_{05}	640	63390	64030	100
D_{11}	8003	0	64030	8
D_{12}	7114	0	64030	9
D_{13}	6403	0	64030	10
D_{14}	704	0	64030	90
D_{15}	640	0	64030	100

Table 2 – The definition of TP, TN, FP, FN.

	predict-phishing	predict-legitimate
real-phishing	TP	FN
real-legitimate	FP	TN

5.1. *Datasets and metrics*

We collect 68030 legitimate URLs from 5000 Best Websites homepage (2012) and 12003 phishing URLs from PhishTank homepage (2019). The phishing URLs have been validated from April 2018 to July 2018. We randomly extract 4000 legitimate URLs and 4000 phishing URLs to form the test dataset. Hence, we can use the left 64030 legitimate URLs and 8003 phishing URLs for training. GAN is adopted to generate phishing URLs in order to make the sum of phishing URLs as 64030 equal to the number of legitimate URLs. Next, with these data, our classifier does phishing detection. Table 1 enumerates the composition of the training dataset. The last column is the ratio between real-world legitimate URLs and real-world phishing URLs. We use D_{nn} to name different datasets. For D_{01}, D_{02} and D_{03}, the ratio between real-world legitimate URLs and real-world phishing URLs is near 7, 8 and 9. For D_{04} and D_{05}, the ratio is about 90 and 100.

We introduce metrics of Accuracy (Acc), FPR, Recall (Rec), Precision (Pre) and F1 score (F1) to evaluate the performance of classifiers. Table 2 gives the definition of TP, TN, FP, FN. Consequently the five metrics can be calculated as follow:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

Acc shows the rate that the classifier correctly detects phishing and legitimate URLs. It intuitively demonstrates the performance of the classifier.

$$FPR = \frac{FP}{FP + TN} \tag{10}$$

FPR is the percentage of real-legitimate but predict-phishing URLs in all real-legitimate URLs. If FPR is high, the

Table 3 – The classification results on different dataset.

Dataset	Acc	FPR	Rec	Pre	F1
D_{01}	97.20%	1.20%	95.60%	98.76%	97.15%
D_{02}	96.76%	0.86%	94.38%	99.10%	96.68%
D_{03}	96.41%	1.23%	94.05%	98.71%	96.32%
D_{04}	92.05%	0.47%	84.57%	99.44%	91.41%
D_{05}	90.54%	0.22%	81.30%	99.73%	89.58%

alert for phishing will be frequent. Hence, a good phishing classifier should have a low FPR.

$$Rec = \frac{TP}{TP + FN} \tag{11}$$

Rec is the rate of real-phishing URLs that we detect in all real-phishing URLs. It is expected to be 1 which means the classifier can detect all phishing URLs.

$$Pre = \frac{TP}{TP + FP} \tag{12}$$

Pre denotes the percentage of real-phishing URLs that we detect in all predict-phishing URLs. Also, it is higher, the performance of the classifier is better.

$$F1 = \frac{2 \times Pre \times Rec}{Pre + Rec} \tag{13}$$

F1 is the harmonic average of Rec and Pre. If the dataset is imbalanced, the classifier tends to recognize all test data as the majority class, i.e., the legitimate class, which achieves a high Acc. But in this case Rec is 0. Hence, in our experiments, Rec is more important than Acc.

5.2. *Results of GAN*

In these experiments of this section, our dataset is composed of 64030 legitimate URLs, some real-world phishing URLs, and generative phishing URLs made by GAN. By running CNN-LSTM on different datasets, we firstly show some examples of generative phishing URLs and then answer the first two research questions below. Table 5 lists some generative phishing URLs using dataset D_{01}.

Q1: Do different ratios between real-world legitimate URLs and real-world phishing URLs impact the classification results?

A1: Since the ratio between real-world legitimate URLs and real-world phishing URLs is unknown and difficult to measure, we do experiments with different ratios and explain the results.

Table 3 presents the classification results with different ratios. The highest Acc, Rec and F1 can achieve 97.20%, 95.60% and 97.15% when the dataset is D_{01}. On the contrary, the lowest Acc, Rec and F1 are 90.54%, 81.30% and 89.58% on D_{05}. The more the real-world phishing URLs are in the training dataset, the higher the Acc is. The trend of Rec and F1 is the same as that of Acc. It is perhaps that the performance of the classifier is better when the training dataset is more balanced. Hence, using the balanced dataset which is composed of real-world phishing URLs and real-world legitimate URLs is helpful for improving classification results. However, in the real world the dataset of phishing URLs and legitimate URLs is not balanced.

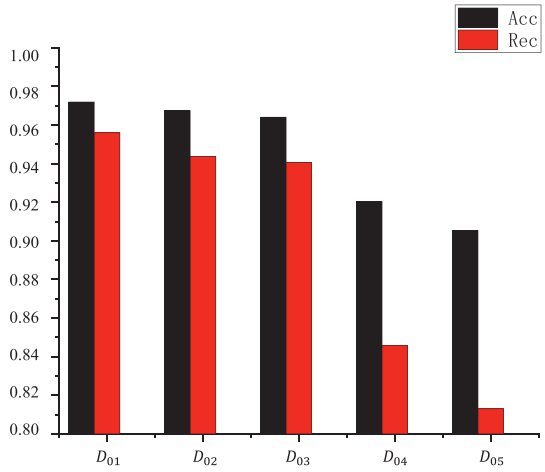


Fig. 10 – Acc, Rec results on different datasets.

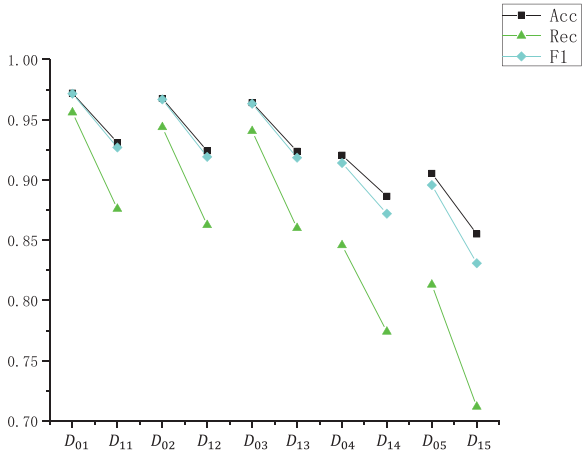


Fig. 11 – Acc, Rec and F1 on different training dataset ($D_{01} - D_{05}$ *vs.* $D_{11} - D_{15}$).

More specifically, Fig. 10 shows the Acc, Rec results on $D_{01} - D_{05}$. From Fig. 10, on different datasets, the variation range of Rec is larger than Acc. It is perhaps that, the classifier tries to classify data into the majority class in order to improve the performance to a certain extent. Hence, for imbalanced training dataset classification, Rec may be more important than Acc.

Q2: How about the experiments on the dataset including phishing URLs generated by GAN and real-world URLs and on the dataset with only real-world URLs?

A2: $D_{01} - D_{05}$ are datasets including real-world URLs and generated phishing URLs, while $D_{11} - D_{15}$ contain only real-world URLs. The results of Acc, Rec and F1 on different training datasets ($D_{01} - D_{05}$ *vs.* $D_{11} - D_{15}$) are given in Fig. 11. Compared with the results of D_{0n} and D_{1n}, Acc, Rec and F1 of D_{0n} are higher than those of D_{1n}. Furthermore, Rec of D_{0n} are higher than that of D_{1n} by about 8%. As we mentioned above, Rec is much more important than Acc. The results illustrate that our classifier performs better with generative phishing URLs

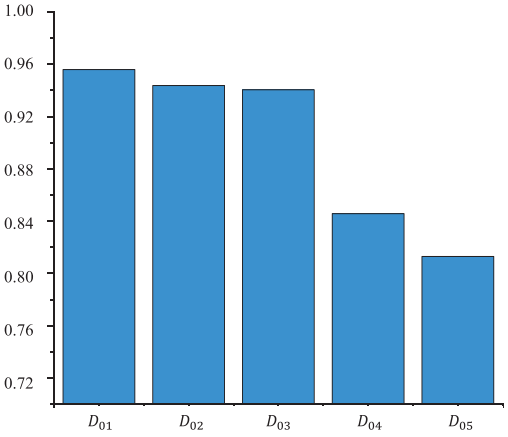


Fig. 12 – Rec on different training datasets (D_{01} – D_{05}).

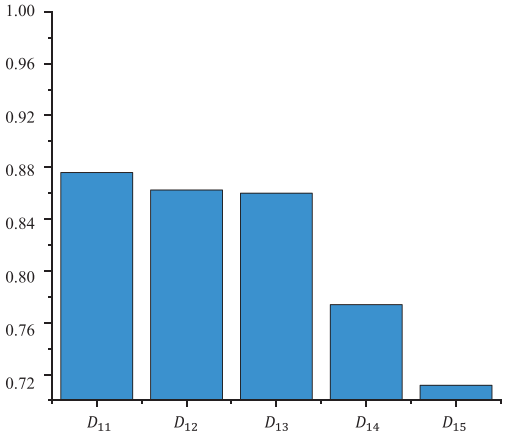


Fig. 13 – Rec on different training datasets (D_{11} – D_{15}).

and the real-world URLs than with only real-world URLs. From that, it can be drawn a conclusion that our generative phishing URLs are helpful for phishing detection.

Furthermore, Fig. 12 plots Rec results on different training datasets (D_{01} – D_{05}) and Fig. 13 shows Rec results on different training datasets (D_{11} – D_{15}). It can be easily seen from these two figures, Rec on D_{01} and D_{11} are the highest among these datasets. Further, if there are more real-world phishing URLs in the dataset, Rec is much higher. In the following, the experiments are conducted on the same dataset with various parameters of self-attention CNN to explain the third question.

Q3: How do parameters influence the performance of our classifier?

A3: *M* and *N* are the numbers of attention blocks and feature blocks, we do experiments with different *M* and *N* to find the impact they bring to the results. We use D_{11} as the training dataset. Table 4 records the results with various *M* and *N*. Acc and Rec both become higher when *M* and *N* increase. If *M* and *N* are both 5, we can get the best results. However, the increase of *M* and *N* may bring huge training costs. It can be seen, when *M* and *N* are bigger than 3, Acc and Rec are almost changeless. That means, *M* and *N* of both 3 may help the classifier get good

Table 4 – The classification results of different *M* and *N*.

The number of attention blocks (*M*)	The number of feature blocks (*N*)	Acc	Rec
1	1	91.32%	85.18%
2	2	92.45%	86.13%
3	3	93.10%	87.6%
4	4	93.12%	87.81%
5	5	93.15%	87.88%

Table 5 – Generative phishing URLs.

http://br486.teste.webse/~casaoba/produto=825153768
http://russia-model2018/id=6445736#gotox
http://www4.sa.irs.tax.gov.us.3066036666.revenue.suptsution.com/sa_www4/start

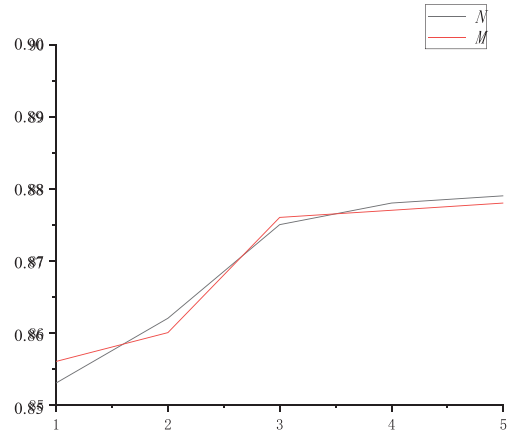


Fig. 14 – Rec with different *M* or *N*.

results, and at the same time take less resources. Hence, we choose *M* and *N* both as 3 to conduct other experiments.

Furthermore, we conduct some experiments with different *M* or *N* on D_{11}. Figure 14 shows Rec with various *M* and fixed *N* of 3, and that with different *N* when *M* is 3. When *M* or *N* increases, Rec becomes higher and higher. This phenomenon suggests that, when the network gets deeper, the performance of the network becomes better. It is perhaps that the deeper network has more parameters than the simple network, which may increase the fitting ability of the network.

5.3. *Comparison of proposed solution with existing anti-phishing approaches*

Since there are many methods to do phishing detection and solve the problem of imbalanced datasets, we compare some methods with our approach and answer the last two research questions here.

Q4: What's the situation when our classifier is compared with the other existing schemes?

Table 6 – The classification results of different methods.

	Acc	FPR	Rec	Pre	F1
GAN(our method)	92.05%	0.47%	84.57%	99.44%	91.41%
SMOTE Chawla et al. (2002)	90.17%	3.6%	84.00%	95.86%	89.52%
conventional method	88.77%	2.10%	79.60%	97.48%	87.64%
imbalanced dataset	88.64%	0.12%	77.40%	99.85%	87.20%

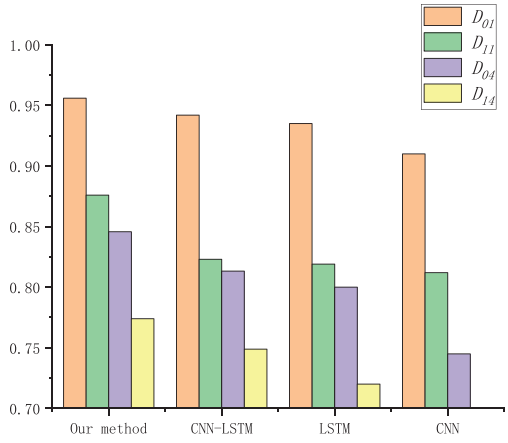


Fig. 15 – The comparison results of Rec between our method and the classic schemes on datasets D_{01}, D_{11}, D_{04} and D_{14}.

Table 7 – The comparison results and detection time with existing anti-phishing approaches.

	Acc	Detection time
Self-attention CNN (our method)	97.20	174μs
Wei et al. (2020)	94.76	84μs
Su (2020)	95.12	34μs

A4: To answer this question, we first validate the superiority of our proposed neural network architecture by comparing with other architectures. Then, we compare our method with existing state-of-the-art anti-phishing approaches (Su, 2020; Wei et al., 2020).

We compare three deep learning networks, i.e., CNN-LSTM, single CNN and single LSTM, with our method (self-attention CNN) in phishing detection. CNN-LSTM is the combination of CNN and LSTM. Figure 15 plots the comparison results of Rec with D_{01}, D_{11}, D_{04} and D_{14}. In the figure, LSTM refers to single LSTM, CNN represents single CNN. It can be seen from the figure, our method achieves higher Rec than the other methods and CNN-LSTM is better than LSTM and CNN regardless of datasets. Perhaps, multi-head self-attention can find the inner dependency relationships between different characters more effective than LSTM (Vaswani et al., 2017). Further it can give different weights to features learnt by CNN so that focus on some important features. The URL representation in our method is comprehensive. Hence, our method for phishing detection can achieve better results. Furthermore, it is evident in Fig. 15 that all methods can get better results on generative phishing URLs and real-world ones than those on only real-world URLs. It points out that our method to get datasets balanced is helpful for phishing detection regardless of the choice of phishing classification methods.

In order to make the results more convincing, we compare the results of the proposed method with existing state-of-the-art anti-phishing approaches (Su, 2020; Wei et al., 2020). The method proposed in Wei et al. (2020) is CNN-based model and Su (2020) detects phishing URLs based on LSTM. Table 7 shows the comparison results and detection time with these approaches on dataset D_{01}. As for the results of *Accuracy*, our method achieves 97.2%, where Wei et al. (2020) and Su (2020) get 94.76% and 95.12% Acc, respectively. The *Accuracy* result of the proposed method is higher than those of Wei et al. (2020) and Su (2020) by 2.44% and 2.08%, respectively. It can be drawn a conclusion from the results that our method of phishing detection can achieve better results than the state-of-the-art methods. In the testing phase, the average detection time of the self-attention CNN is 174μs. As for the method proposed by Wei et al. (2020), the average time to process and detect one URL sample is 84μs which is 3 times faster than our method. The average detection time of the approach proposed by Su (2020) is 34μs, which is 2 times faster than the method proposed by Wei et al. (2020). Compared with the proposed self-attention CNN, Wei et al. (2020) and Su (2020) utilize simpler neural networks with less computations and thus spends less time. Let n be the sequence length, d be the hidden dimension, k be the kernel size of the convolution layer. In the case of the proposed self-attention CNN, we have $n = 99$, $d = 64$, $k = 3$ and the number of the self-attention and the convolution layers are both 3. According to the Table 1 in the paper proposed by Vaswani et al. (2017), the computation complexity per layer of Self-Attention, Recurrent (i.e., LSTM), and Convolution are $O(n^2 \cdot d)$, $O(n \cdot d^2)$, and $O(k \cdot n \cdot d^2)$, respectively. So theoretically our model is a little more complex, and the experimental results show the same conclusion. As for the detection time, even if our method is the slowest, our method takes only 174μs (i.e., 1.74×10^{-4}s) to detect one website. In other words, our method can detect 5,747 URL samples per second. In general, our method gets higher *Accuracy* while keeping acceptable detection speed.

Q5: Are generative URLs created by GAN more useful for classification than those made by the other methods?

A5: We compare some approaches for solving the imbalanced dataset problem with our method, including SMOTE and the conventional method. SMOTE (Chawla et al., 2002) takes a random point on the line between the samples of the minority class and their adjacent samples to generate new samples of the minority class. And the conventional method to deal with the imbalanced dataset problem is to copy the samples in the minority class some times, leading to the number of these samples equal to that of the samples in the majority class. We use our method (GAN), SMOTE and the conven-

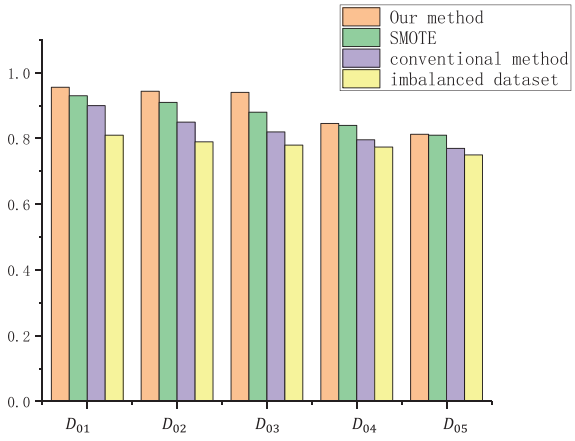


Fig. 16 – The comparison results of Rec between our method and the classic schemes on datasets D_{01} – D_{05}.

tional method to generate phishing URLs. In this experiment, our original dataset includes 704 phishing URLs randomly extracted from our training dataset and 64030 legitimate URLs. It is an imbalanced dataset. In the classification phase, the same method, self-attention CNN, is utilized.

Table 6 lists the three classification results on the balanced dataset created by the above three methods and the classification result on the original imbalanced dataset. It is easy to see from the table that our method gets the highest Acc, Rec and F1. Moreover, experiments with different numbers of real-world phishing URLs are conducted, the Rec results of which are shown in Fig. 16. On all datasets, the rank list of Rec according to the descend order is our method, SMOTE, the conventional method, the imbalanced dataset. It demonstrates that the generative phishing URLs by our method are much more effective for phishing detection than those generated by the other methods. Maybe it is because GAN can learn features exactly and use them to create phishing URLs so that these URLs are more similar with real-world phishing URLs. More specifically, attackers may follow certain rules to product phishing URLs, and the rules can be learnt by GAN. From Fig. 16, there is another indication that the dataset with more real-world phishing URLs helps the classifier get higher Rec.

6. Conclusion and future works

In order to identify phishing websites with a high-accuracy rate, in this paper, we propose self-attention CNN, a CNN and multi-head self-attention combined deep learning approach. We first introduce GAN to generate phishing URLs, making the training dataset balanced. Then the new deep network involving CNN and multi-head self-attention is built to do phishing websites detection. Experiments show that the classifier on the dataset of generative URLs and real-world URLs can achieve better classification results than on that of only real-world URLs. The phishing URLs produced by GAN are more effective for phishing websites detection than those by the other method. Furthermore, we also validate the superiority of our proposed neural network architecture. The proposed self-attention CNN achieves higher performance than other deep learning network architectures, i.e., CNN-LSTM, single CNN and single LSTM. Compared with the existing anti-phishing approaches, our proposed method outperforms the state-of-the-art methods (Su, 2020; Wei et al., 2020) by a significant margin.

In terms of the limitation, the proposed approach does not consider the features of the HTML content. Therefore, in the future, we will utilize HTML content of websites to enrich the features we use. As for the detection time, we will optimize self-attention CNN with less computation resource in the future work. Possible optimizations include sequence-level knowledge distillation techniques (Kim et al., 2019). Meanwhile, the dataset used in this work is collected in 2018. We think the pattern of phishing sites now is similar to that of 2018. In the future work, we will collect a new dataset and try to perform comparative experimental analysis on these two datasets simultaneously to exploring the evolution pattern of phishing websites over time.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Xi Xiao: Conceptualization, Methodology. **Wentao Xiao:** Writing - original draft. **Dianyan Zhang:** Investigation, Writing - original draft. **Bin Zhang:** Project administration. **Guangwu Hu:** Writing - review & editing. **Qing Li:** Visualization, Validation. **Shutao Xia:** Project administration.

REFERENCES

5000 BEST WEBSITES homepage, http://5000best.com/websites/; 2012.

Abutair H, Belghith A. Using case-based reasoning for phishing detection. Procedia Comput. Sci. 2017;109:281–8. doi:10.1016/j.procs.2017.05.352.

Aravindhan R, Shanmugalakshmi R, Ramya K, Chinnaiyan S. Certain investigation on web application security: phishing detection and phishing target discovery; 2016. p. 1–10.

Blum A, Wardman B, Solorio T, Warner G. Lexical feature based phishing URL detection using online learning; 2010. p. 54–60.

Chan P., Stolfo S.. Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection1998;.

Chawla N, Bowyer K, Lawrence OH, Philip Kegelmeyer W. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res (JAIR) 2002;16:321–57. doi:10.1613/jair.953.

Chen W., Xu A.W., Wei Z., Xu C.. Phishing detection research based on PSO-BP neural network2018;.

Chen Xw, Gerlach B, Casasent D. Pruning support vectors for imbalanced data classification, vol. 3; 2005. p. 1883–8.

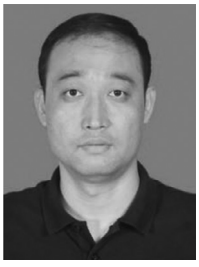
Corona I, Biggio B, Contini M, Piras L, Corda R, Mereu M, Mureddu G, Ariu D, Roli F. DeltaPhish: detecting phishing webpages in compromised websites; 2017. p. 370–88.

Correa Bahnsen A, Contreras Bohorquez E, Villegas S, Vargas J, González F. Classifying phishing URLs using recurrent neural networks; 2017. p. 1–8.
Cui Q, Jourdan GV, Bochmann G, Couturier R, Onut IV. Tracking phishing attacks over time; 2017. p. 667–76.
Dunham K. Mobile malware attacks and defense; 2009. doi:10.1016/B978-1-59749-298-0.X0001-8.
Futai Z, Yuxiang G, Bei P, Li P, Linsen L. Web phishing detection based on graph mining; 2016. p. 1061–6.
Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. ArXiv 2014.
Gowtham R, Gupta J, Gamya PG. Identification of phishing webpages and its target domains by analyzing the feign relationship. J. Inf. Secur. Appl. 2017;35:75–84. doi:10.1016/j.jisa.2017.06.001.
Gowtham R, Krishnamurthi I. A comprehensive and efficacious architecture for detecting phishing webpages. Comput. Secur. 2014;40:2337. doi:10.1016/j.cose.2013.10.004.
He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition; 2016. p. 770–8.
Jain A, Gupta BB. Towards detection of phishing websites on client-side using machine learning based approach. Telecommun. Syst. 2017;68. doi:10.1007/s11235-017-0414-0.
Ketkar N.. Convolutional neural networks2017;.
Kim HG, Na H, Lee H, Lee J, Kang TG, Lee MJ, Choi YS. Knowledge distillation using output errors for self-attention end-to-end models. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2019. p. 6181–5. doi:10.1109/ICASSP.2019.8682775.
Kim Y.. Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing2014;. 10.3115/v1/D14-1181
Le A, Markopoulou A, Faloutsos M. PhishDef: URL names say it all. Proceedings - IEEE INFOCOM, 2010.
Lecun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436.
Liang B, Su M, Wei Y, Shi W, Gang Y. Cracking classifiers for evasion: a case study on the google's phishing pages filter. International Conference on World Wide Web, 2016.
Lin Z., Feng M., Dos Santos C., Yu M., Xiang B., Zhou B., Bengio Y.. A structured self-attentive sentence embedding2017;.
Ludl C, McAllister S, Kirda E, Kruegel C. On the effectiveness of techniques to detect phishing sites; 2007. p. 20–39.
Ma J, Lawrence KS, Savage S, Geoffrey MV. Beyond blacklists: learning to detect malicious web sites from suspicious URLs; 2009. p. 1245–54.
Marchal S, Francois J, State R, Engel T. PhishStorm: detecting phishing with streaming analytics. IEEE Trans. Netw. Serv. Manage. 2014;11:458–71. doi:10.1109/TNSM.2014.2377295.
Marchal S, Saari K, Singh N, Asokan N. Know your phish: Novel techniques for detecting phishing sites and their targets; 2016. p. 323–33.
Mcafee phishing protection, https://home.mcafee.com/advicecenter/?id=ad_phishing&ctst=1; 2019a.
Mohammad R, Thabtah F, Mccluskey T. An assessment of features related to phishing websites using an automated technique; 2012. p. 492–7.
Money Harris D., Sarah L.H.. Digital design and computer architecture2007;. 10.1016/B978-0-12-370497-9.X5000-8
Panda security, http://www.pandasecurity.com/canada-eng/homeusers/security-info/cybercrime/phishing/; 2019b.
Peng Y, Guangzhen Z, Peng Z. Phishing website detection based on multidimensional features driven by deep learning. IEEE Access 2019;7:15196–209. doi:10.1109/ACCESS.2019.2892066.
Phishtank homepage, https://www.phishtank.com; 2019.
Rajab M. An anti-phishing method based on feature analysis; 2018. p. 133–9.
Report from APWG, http://docs.apwg.org/reports/apwg_trends_report_q4_2018.pdf; 2018.
Rsa online fraud report, https://www.rsa.com/en-us/perspectives/industry/online-fraud; 2019.
Schölkopf B, Platt J, Shawe-Taylor J, Smola A, Robert CW. Estimating support of a high-dimensional distribution. Neural Comput. 2001;13:1443–71. doi:10.1162/089976601750264965.
Srinivasa Rao R, Roshan Pais A. Detection of phishing websites using an efficient feature-based machine learning framework. Neural Comput. Appl. 2018. doi:10.1007/s00521-017-3305-0.
Su Y. Research on website phishing detection based on LSTM RNN, vol. 1; 2020. p. 284–8. doi:10.1109/ITNEC48623.2020.9084799.
Tan CL, Leng Chiew K, Wong K, Nah Sze S. PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder. Decis. Support Syst. 2016;88. doi:10.1016/j.dss.2016.05.005.
Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Aidan N.G., Kaiser L., Polosukhin I.. Attention is all you need2017;.
Verma R, Das A. What's in a URL: fast feature extraction and malicious URL detection. In: Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics. New York, NY, USA: ACM; 2017. p. 55–63. http://doi.acm.org/10.1145/3041008.3041016. doi:10.1145/3041008.3041016.
Wei W, Ke Q, Nowak J, Korytkowski M, Scherer R, Woźniak M. Accurate and fast URL phishing detector: a convolutional neural network approach. Comput. Netw. 2020;178:107275. doi:10.1016/j.comnet.2020.107275. https://www.sciencedirect.com/science/article/pii/S1389128620301109
Wenyin L, Liu G, Qiu B, Quan X. Antiphishing through phishing target discovery. IEEE Internet Comput. - INTERNET 2012;16:52–61. doi:10.1109/MIC.2011.103.
Zhang D, Yan Z, Jiang H, Kim T. A domain-feature enhanced classification model for the detection of chinese phishing e-business websites. Inf. Manage. 2014;51. doi:10.1016/j.im.2014.08.003.
Zhang H, Liu G, Chow T, Wenyin L. textual and visual content-based anti-phishing: a bayesian approach. IEEE Trans. Neural Netw. 2011;22:1532–46. doi:10.1109/TNN.2011.2161999.
Zhang J., Li X.. Phishing detection method based on borderline-smote deep belief network2017;.
Zhang J, Phillip AP, Ullrich J. Highly predictive blacklisting; 2008. p. 107–22.
Zhang Y, Hong J, Cranor L. CANTINA: a content-based approach to detecting phishing web sites; 2007. p. 639–48.
Zuhair H, Selamat A. Phishing classification models: issues and perspectives. Int. J. Digit. Enterprise Technol. 2019;1:219. doi:10.1504/IJDET.2019.097845.

Xi Xiao is an associate professor in Graduate School at Shenzhen, Tsinghua University. He got his Ph.D. degree in 2011 in State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences. His research interests focus on information security and the computer network.

Wentao Xiao is pursuing his Master degree in computer technology at University of Tsinghua. His research interests focus on machine learning, deep learning, and cyberspace security.

Dianyan Zhang is pursuing his Master degree in computer technology at University of Tsinghua. His research interests focus on network security and deep learning.



Bin Zhang received his Ph.D. degree in Department of Computer Science and Technology, Tsinghua University, China in 2012. He worked as a post doctor in Nanjing Telecommunication Technology Institute from 2014 to 2017. He is now a researcher in the Cyberspace Security Research Center of Peng Cheng Laboratory. His current research interests focus on network anomaly detection, Internet architecture, and its protocols, network traffic measurement, information privacy security, etc.

Guangwu Hu is an associate professor of Shenzhen Institute of Information Technology, China. He received his Ph.D. degree in computer science and technology from Tsinghua University in 2014. His research interests include software defined networking, Next-Generate Internet and Internet security.

Shutao Xia is a professor and a Ph.D. supervisor in Graduate School at Shenzhen, Tsinghua University. He got his Ph.D. degree in Nankai University 1997. He mainly engaged in information theory and coding, internet and big data. He has published more than 60 papers in top international journals and international conferences.