2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

Phishing URL Detection via CNN and Attention-Based Hierarchical RNN

Yongjie Huang, Qiping Yang, Jinghui Qin, Wushao Wen*
School of Data and Computer Science
Sun Yat-Sen University
Guangzhou, China
Email: {huangyj45, yangqp5, qinjingh}@mail2.sysu.edu.cn, wenwsh@mail.sysu.edu.cn
* Corresponding Author

***Abstract*—Phishing websites have long been a serious threat to cyber security. For decades, many researchers have been devoted to developing novel techniques to detect phishing websites automatically. While state-of-the-art solutions can achieve superior performances, they require substantial manual feature engineering and are not adept at detecting newly emerging phishing attacks. Therefore, developing techniques that can detect phishing websites automatically and handle zero-day phishing attacks swiftly is still an open challenge in this area. In this work, we propose PhishingNet, a deep learning-based approach for timely detection of phishing Uniform Resource Locators (URLs). Specifically, we use a Convolutional Neural Network (CNN) module to extract character-level spatial feature representations of URLs; meanwhile, we employ an attention-based hierarchical Recurrent Neural Network(RNN) module to extract word-level temporal feature representations of URLs. We then fuse these feature representations via a three-layer CNN to build accurate feature representations of URLs, on which we train a phishing URL classifier. Extensive experiments on a verified dataset collected from the Internet demonstrate that the feature representations extracted automatically are conducive to the improvement of the generalization ability of our approach on newly emerging URLs, which makes our approach achieve competitive performance against other state-of-the-art approaches.**

***Index Terms*—Phishing Detection, Cyber Security, Machine Learning, Deep Learning**

I. Introduction

According to Anti-Phishing Working Group (APWG)[1], a not-for-profit industry association focused on eliminating the identity theft and frauds that result from the growing problem of phishing, crimeware, and email spoofing, phishing is a criminal mechanism which employs both social engineering and technical subterfuge to steal customers' personal identity information and financial account data. Recent years have witnessed the rapid growth of phishing attacks. As reported by APWG [1], the number of detected phishing attacks in the first half of the year 2018 was 496,578, compared to 371,519 in the second part of 2017, resulting in over 33% growth in half a year. Phishing has become a severe problem because of the serious damage caused to its targeted industry, e.g., payment, financial institution, email, etc. Estimates of annual direct economic loss to the American economy due to phishing attacks are between 61 million dollars and 3 billion dollars [2].

[1] https://www.antiphishing.org/

To mitigate the threat of phishing, researchers have been working on improving the accuracy of phishing detection via various list-based and machine learning-based methods that leverage handcrafted features of URLs, host information and website contents [3]. Nevertheless, phishing detection remains an arms race with no definitive solution due to the constant evolution of phishing, which makes techniques that can extract useful features automatically and detect phishing websites accurately to thwart phishing attacks continue to be a pressing need.

To address this problem, we propose a deep learning-based approach which can identify phishing websites accurately via the feature representations extracted automatically from URLs. According to Hong [2], phishing attacks can spread by embedding phishing URLs into spoofed messages. Therefore, developing techniques to detect phishing URLs effectively can help to thwart phishing attacks to a great extent. The reason why we extract feature representations from URLs only is twofold. First, extracting features from host information or website contents is time consuming because of the necessary interaction with the websites. Second, URL-based phishing detection [4], [5] has shown promising performance since it not only is lightweight but also has relatively high detection accuracy. Because of the focus on the URL itself, our approach can detect phishing websites faster than other methods that consider the website contents or host information additionally; meanwhile, our approach can be applied to anywhere that a URL can be embedded, e.g., emails, twitter messages, websites, etc.

In this paper, we focus on detecting phishing websites using only their URLs via deep learning techniques. Specifically, we use a CNN module to extract character-level spatial feature representations of URLs; meanwhile, we employ an attention-based hierarchical RNN module to extract word-level temporal feature representations of URLs. To make it clear, we first utilize one bidirectional Long Short-Term Memory (LSTM) layer to generate the intermediate character-level temporal features from characters, and then we use another one to extract word-level temporal feature representations from the intermediate character-level temporal features. Subsequently, we fuse these feature representations together via a three-layer CNN, and then use a Multi-Layer Perceptron (MLP) to identify the phish-

2324-9013/19/$31.00 ©2019 IEEE
DOI 10.1109/TrustCom/BigDataSE.2019.00024

ing websites. To evaluate the performance of our approach, extensive experiments were conducted on a dataset comprised of $4{,}820{,}940$ URLs, among which $241{,}047$ are phishing URLs from both Phishtank[2] and Openphish[3] while the others are legitimate ones from Alexa[4]. We compare our approach with seven state-of-the-art methods, which can be categorized into three groups, namely heuristic-based, machine learning-based and deep learning-based detection respectively. Since we have constructed an imbalanced dataset where the ratio of legitimate websites to phishing ones is 95:5 to represent the real world distribution as precise as possible, and Area Under the Receiver Operating Characteristic Curve (AUC) [6] is insensitive to such imbalanced dataset, we consider AUC together with accuracy, false positive rate, and precision as our evaluation metrics to compare the performances of different methods in phishing URLs detection. Experimental results show that our approach outperforms the state-of-the-art methods across all the evaluation metrics. In particular, the AUC of our approach can reach 0.99269 while the highest AUC of the state-of-the-art methods is 0.95515, which demonstrates the effectiveness of the proposed approach.

Our primary contributions include:

- We propose a novel neural network consisting of a CNN and an attention-based hierarchical RNN for phishing URL detection. The spatial and temporal feature representations of URLs extracted by our approach automatically are proved to be effective according to the experimental results.
- We propose a novel objective function and conduct extensive experiments on the large validated dataset we collected with the proposed objective function. The experimental results demonstrate the effectiveness and the efficiency of our approach, indicating the superior performance and generalization ability of our approach.

The remainder of this paper is organized as follows: In Section II, we will review the representative works on phishing detection. Next, we shed light on the details of our proposed approach in Section III. Subsequently, in Section IV, we will describe our experimental setup and analyze the experimental results. Finally, conclusion will be provided in Section V.

II. Related Work

Generally, phishing detection can be done via list-based, heuristic-based, machine learning-based or deep learning-based methods. However, the phishing problem is so complex that no definitive solution exists to eliminate all the threats effectively; thus, multiple techniques are often developed to thwart specific attacks.

A. List-Based Detection

List-based phishing detection algorithms can be categorized into whitelist-based and blacklist-based schemes. Wang et al. [7] proposed to detect phishing websites with a whitelist technique, which was claimed to be lightweight and efficient. Blacklist-based approaches are used widely in publicly available anti-phishing toolbars, e.g., Google Safe Browsing[5], which checks URLs against Google's constantly updated blacklist of phishing websites on browsers and gives warnings to the users once it judges a URL as a phishing one. Though such list-based methods can have a relatively high precision, as new URLs are generated on a day to day basis, it is difficult to retain a comprehensive list of phishing URLs. Prakash et al. [8] proposed an improved list-based method with two components to prevent phishers from evading the blacklist detection. While one of the components is exploited to expand the blacklist through simple combinations of phishing sites from the blacklist using five heuristics (i.e., top-level domain, IP address, directory structure, query string, brand name), the other one is used to perform an approximate match of a given URL to determine whether it is a phishing one. However, list-based approaches suffer from an inability to defend against zero-day attacks, which means that they cannot react to the newly emerging phishing URLs until developers update the list.

B. Heuristic-Based Detection

The core of heuristic-based approaches, which are developed from list-based schemes, is to generate the characteristics of phishing websites based on several handcrafted features, e.g., URL-based lexical features, URL-based host features, webpage contents, website visual similarity, etc. Zhang et al. [9] proposed an approach named Cantina to detect phishing websites based on webpage contents. By exploiting various features from HTML Document Object Models (DOMs), search engines and third party services, they managed to improve the detection accuracy. Specifically, they extracted key terms of a webpage using Term Frequency-Inverse Document Frequency (TF-IDF) to provide a unique signature of the webpage. After supplying this signature to a search engine (e.g., Google), Cantina infers the legitimacy of a webpage. Another representative approach called Cantina+ was proposed by Xiang et al. [10]. As an enhanced version of Cantina, Cantina+ takes 15 heuristic features as input to train a classifier for phishing websites detection. Yet, despite the improvement of detection accuracy, unfortunately, manually exploiting potentially useful features not only is time consuming but also requires constant adaptation to evolving evading techniques, which imposes a restriction on the achievable performances of the detectors.

C. Machine Learning-Based Detection

As pointed out by Sahoo et al. [11], while extra information can improve phishing detection accuracy (subject to appropriate regularization), it is often not practical to obtain a lot of features due to the limited time and available computation resource. To address these limitations, recent years have witnessed several efforts to detect phishing websites with machine

[2]http://phishtank.com/
[3]https://openphish.com/
[4]https://www.alexa.com/
[5]https://developers.google.com/safe-browsing/

learning techniques. Cui et al. [12] proposed to detect phishing websites via a hierarchical clustering approach which groups the vectors generated from DOMs together according to their proportional distance. Apart from clustering, many researchers focus on detecting phishing websites through classification algorithms in machine learning. Zhang et al. [13] extracted features from URLs based on bag-of-words and then trained a classifier on the features via online learning. Similarly, in another work, Sahingoz et al. [14] adopted distributed representations of words within a given URL, and then used seven different machine learning algorithms to predict whether it was a phishing URL. Albeit the above approaches have shown satisfying performances, they suffer from the following limitations: (i) Inability to handle unseen characteristics: Since the URLs to be detected are likely to contain unknown words that do not exist in the training set, word-based methods might be less powerful under this circumstance; (ii) Loss of semantic information: Bag-of-words model cannot retain the semantic information since word order is ignored.

D. Deep Learning-Based Detection

To address the aforementioned defects, some deep learning-based phishing website detection solutions have recently come to light due to the success in Natural Language Processing (NLP) achieved by deep learning. Some researches [4], [5], [15] focused on detecting phishing URLs by leveraging the potential characteristics of URLs. In contrast, other studies [16]–[18] fully exploited content-based features or event-based features for phishing websites detection. The primary difference of our approach w.r.t. the previously cited deep learning-based ones is that we combine the advantages of CNN and attention-based hierarchical RNN to extract novel character-level spatial and word-level temporal feature representations of URLs automatically, which turn out to be conducive to the improvement of phishing detection performance.

III. Proposed Approach

A. Problem Definition

Since the goal of phishing URL detection is to determine whether a URL is phishing, we formulate this problem as a binary classification problem with URLs as input. More formally, for URL u in URL space S, we denote s as the character sequence of u by $[c_1, c_2, \cdots, c_n] \in S$. We classify u as a phishing URL or a legitimate one by training a network $f_\theta : S \rightarrow \{0, 1\}$ that takes as input s and outputs 0 if u is classified as a legitimate URL and 1 otherwise.

B. Model

To detect phishing URLs effectively, we propose PhishingNet, a deep learning-based framework consisting of a CNN module and a hierarchical attention-based RNN module. According to Le et al. [19], features generated from URLs are powerful enough to discriminate phishing URLs from legitimate ones. Besides, the aforementioned studies [4], [5], [15] have demonstrated the viability of deploying deep learning on URLs to detect phishing websites. Therefore, we detect phishing websites based solely on URLs. Considering that extracting feature representations of URLs from the comprising words directly might encounter out-of-vocabulary (OOV) words, and the advantages of character-level CNN shown by Kim et al. [20] and Xiang et al. [21], we adopt CNN to extract character-level spatial feature representations of URLs from the comprising characters. Additionally, we design an attention-based RNN module to extract word-level temporal feature representations of URLs for the reason that RNN and its variants have proved to be good at handling sequential data [22] while a URL is a sequence of characters essentially. The structure of PhishingNet is illustrated in Fig. 1.

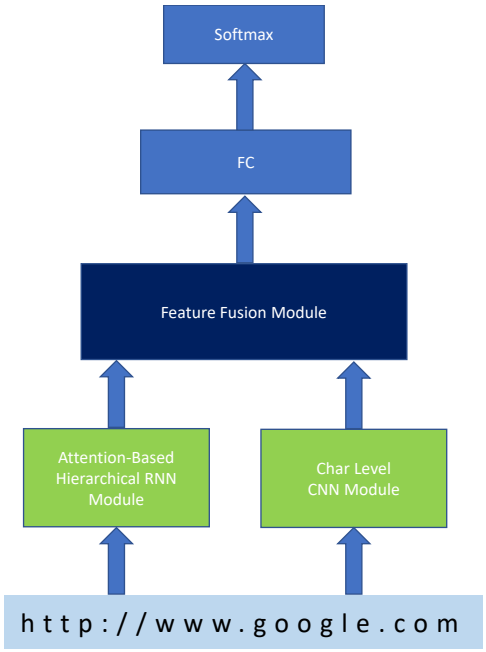


Fig. 1. PhishingNet

We use two parallel modules to extract feature representations of URLs, one of which is a character-level CNN module and the other is an attention-based hierarchical RNN module. The details of these modules will be provided later. After both of these modules finish extracting feature representations from URLs, we utilize a feature fusion module to fuse the retrieved feature representations for generating more effective feature representations of URLs. We adopt three convolutional layers in the feature fusion module to fuse the spatial and temporal feature representations from the character-level CNN module and the attention-based hierarchical RNN module. The feature fusion module is followed by three fully connected layers, finally leading to the output classifier. Below we explain how each of these modules contributes to our overall approach detailedly.

The structure of the attention-based hierarchical RNN module is depicted in Fig. 2. First, we transform a URL into a sequence of words. Specifically, we drop all non-alphanumeric

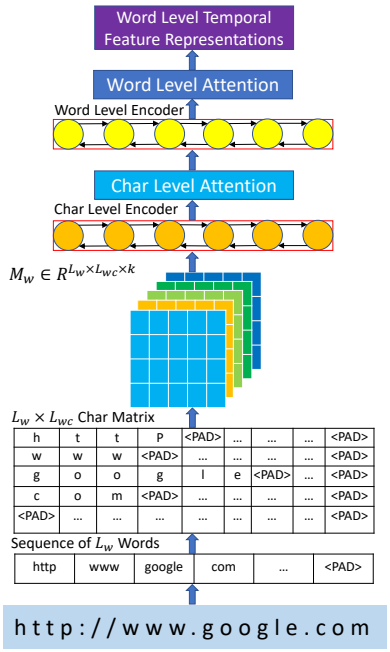


Fig. 2. Attention-Based Hierarchical RNN

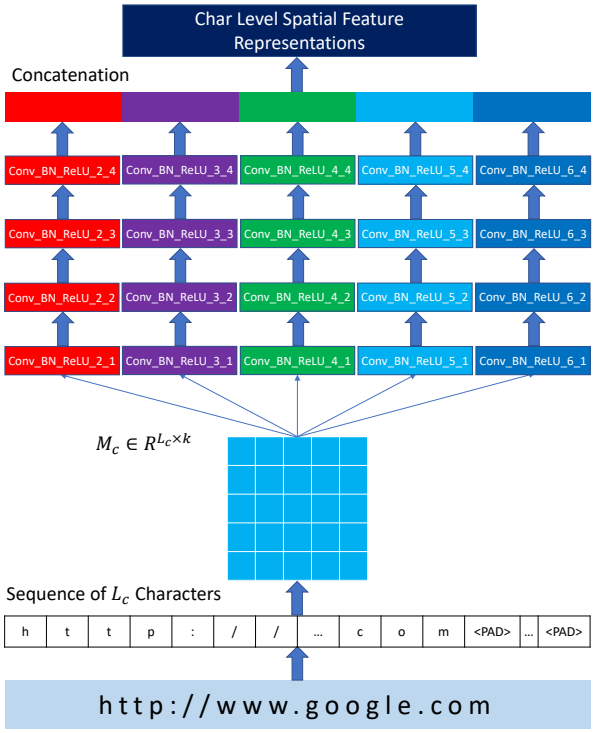


Fig. 3. Character-Level CNN

characters(e.g., ':', ',', '_', etc.) in this step, whereafter we split every word into a sequence of characters. By means of padding or trimming, we transform each URL into a sequence of L_w words, where each word is composed of L_{wc} characters. Through this, we obtain a matrix representation of a URL, where each row is a character sequence of a word in the URL. After that, we copnvert this matrix to a matrix $M_w \in R^{L_w \times L_{wc} \times k}$ via the k-dimensional character-level embedding explained in Section IV-A3. Next, we use a bidirectional LSTM layer equipped with attention mechanism to extract character-level temporal features from M_w, and then employ another to extract word-level temporal feature representations from the character-level temporal features.

In Fig.3, an overview of the character-level CNN module is presented. In this module, we first transform a URL into a sequence of L_c characters through padding or trimming. We then get the distributed representation $M_c \in R^{L_c \times k}$ of the sequence from the k-dimensional character-level embedding. Next, the distributed representation M_c is input to five parallel convolutional blocks where each of them contains four convolutional layers equipped with batch normalization and ReLU activation. Each convolutional block has different kernel sizes for multi-scale learning. Finally, the retrieved features from all convolutional blocks are concatenated as the character-level spatial feature representations.

C. Objective Function

In practice, we found that training our model with the traditional cross-entropy objective function on our imbalanced dataset might let our model be prone to classify many legitimate URLs as phishing ones due to the complexity of our legitimate URL dataset, and thus result in relatively low detection accuracy, high false positive rate, and low precision. Therefore, we propose a novel objective function $\mathcal{L}$ which consists of an optimized cost-sensitive cross-entropy objective function $\mathcal{L}_1$ and the objective function $\mathcal{L}_2$ proposed by Yan et al. [23] to improve the performance and the generalization ability of our approach.

More detailedly, we first design a cost-sensitive cross-entropy objective function with a dynamic cost of misclassification:

$$\mathcal{L}_1(p, y) = -(\alpha y \log(p) + \beta(1-y)\log(1-p)), \quad (1)$$

where p is the predicted probability value for legitimate class, and y is the corresponding label in one-hot representation. Especially, we use α and β to adjust the penalty for misclassification dynamically. To be specific, in (2), n_{all} is the number of samples in a mini-batch, while n_0 and n_1 are the numbers of legitimate samples and phishing samples in that batch respectively. To make it robust, if n_0 or n_1 is 0, we replace n_{all}, n_0, n_1 with N_{all}, N_0, N_1 respectively, where the latter three are the corresponding numbers in the training set:

$$[\alpha, \beta] = \begin{cases} [\frac{n_{all}}{2n_0}, \frac{n_{all}}{2n_1}] & : n_0 \neq 0 \ \wedge \ n_1 \neq 0 \\ [\frac{N_{all}}{2N_0}, \frac{N_{all}}{2N_1}] & : \text{otherwise} \end{cases} \quad (2)$$

The intuition behind the dynamic cost formulation is that the ratio of legitimate URLs to phishing ones in a mini-

batch could fluctuate severely (i.e., the ratio might range from 0:100 to 100:0). Therefore, the cost of misclassification should be adjusted dynamically to optimize the performance of our approach.

To further improve the performance of our approach, we also adopt the objective function $\mathcal{L}_2$ proposed by Yan et al. [23]:

$$\mathcal{L}_2 = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} R(x_i, y_j), \tag{3}$$

$$R(x_i, y_j) = \begin{cases} (-(x_i - y_j - \gamma))^p & : x_i - y_j < \gamma \\ 0 & : \text{otherwise} \end{cases}, \tag{4}$$

where $\{x_0, x_1, \cdots, x_{m-1}\}$ are the classifier outputs for phishing examples, while $\{y_0, y_1, \cdots, y_{n-1}\}$ are the outputs for legitimate examples. Here $0 < \gamma \leq 1$ and $p > 1$.

Therefore, the training objective function of our proposed approach is:

$$\mathcal{L} = \theta \mathcal{L}_1 + (1 - \theta)\mathcal{L}_2, \tag{5}$$

where $\theta \in [0, 1]$.

IV. Experiments

A. Data

1) Data Collection: To train our model and evaluate its performance, we built a dataset containing both legitimate and phishing URLs.

The collection process lasted for three months from August 2018 to November 2018, resulting in a dataset of 4.8 million URLs. Among the data, 5% are phishing URLs collected from PhishTank and Openphish while the other 95% are legitimate ones crawled based on the Alexa top one million sites. The URLs in the Alexa top one million sites are base domain URLs, which are simpler than phishing URLs usually. However, most URLs accessed in the real world are full path URLs which are much more complicated than base domain URLs. Therefore, to simulate the real world situation, we crawled URLs from the Alexa top one million sites, and randomly selected a subset of them as our legitimate URL dataset, which makes our dataset much more complicated than most datasets used in the previous works, e.g., [12], etc. Specifically, we crawled 20 URLs at each depth with a total of five crawl depths, which meant that we crawled 100 full path URLs for each base domain URL collected from Alexa. After that, we randomly selected a subset of these URLs to mix with the phishing ones. To guarantee that the collected URLs were labeled correctly, we used the service provided by VirusTotal[6] for verification and filtered out the mislabeled ones to remove noise. To evaluate the generalization ability precisely, our proposed model is trained on the previous existing URLs and tested on the newly emerging ones. More specifically, we sorted the verified dataset by collection time in ascending order and then split it into three parts, namely training set, validation set, and test set respectively. Other details about the dataset can be found in Table I.

[6]https://www.virustotal.com/

TABLE I
URL Dataset

	Phishing	Legitimate	Total
Training	167,346	3,179,574	3,346,920
Validation	9,905	188,195	198,100
Testing	63,796	1,212,124	1,275,920
Total	241,047	4,579,893	4,820,940

TABLE II
URL Example

URL	Segmentable Words	Segmented Words
http://www.example.com/aboutus/policies	aboutus	<about, us>
http://example.com/LoginMember.htm	LoginMember	<Login, Member>
http://example.com/updateaccount/	updateaccount	<update, account>
http://example.com/reg/emailaccess.html	emailaccess	<email, access>
http://www.updatepaypal.example.com/	updatepaypal	<update, paypal>

2) Data Preprocessing: In essence, a URL is a sequence of characters or words wherein some words have few semantic meanings. What's worse, some URLs contain segmentable joined words without separators, making it harder to extract semantic meanings from words in URLs since these joined words may be meaningless. Example URLs can be referred at Table II. Since the syntax of URLs is complicated, an effective word segmentation algorithm is needed to extract semantic features of words in URLs. To address this issue, we design a word segmentation algorithm utilizing Viterbi [24]. Specifically, we extract domain names of the URLs collected from Common Crawl[7] to build a domain name vocabulary for the reason that domain name is a tried-and-true feature adopted by many previous studies [8]–[10], [25] in phishing detection task. We then merge it with another vocabulary composed of common English words in a ratio of 1:1. In the next step, we obtain a sequence of tokens from a URL with the regular expression pattern `[a-zA-Z]+|[0-9]+`. With the merged vocabulary, we adopt Viterbi to segment the tokens and finally acquire a sequence of segmented words. By using our designed segmentation algorithm instead of segmenting words based on non-alphanumeric characters, we can scale down the number of unique words since many segmentable rare words are segmented to common words. Refer to Algorithm 1 and Algorithm 2 for more details.

3) Embedding: Publicly available word embeddings are not suitable for this task since they are trained on human-readable corpora (e.g., Word2vec [26] takes as its input a large corpus of News, Global Vectors for Word Representation (GloVe) [27] is trained on five corpora varying from Wikipedia to web data, etc.). However, URLs are different from these corpora because they are not always readable for human beings. Based on this insight and the fact that there is an enormous amount of rare words in URLs, we managed to build a k-dimensional character-level embedding which took URLs as its input. Specifically, we trained our k-dimensional character-level embeddings using the Skip-gram model introduced by Mikolov et al. [26] with a total of $3{,}295{,}473{,}093$ unclassified

[7]http://CommonCrawl.org/

Algorithm 1: Vocabulary Merge

Input:
Common words vocabulary V_c, domain name vocabulary V_d
Common words frequency C_c, domain name frequency C_d
Output:
Final vocabulary set V, probability set P

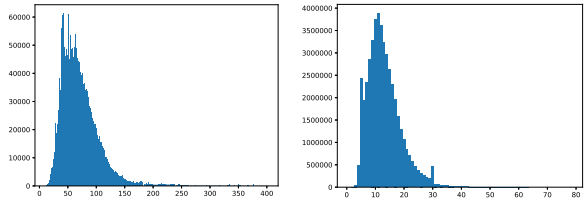
$w_1 \leftarrow 0.5$;
$w_2 \leftarrow 0.5$;
$sum_c \leftarrow \sum C_c$;
$sum_d \leftarrow \sum C_d$;
$V \leftarrow$ empty set;
$P \leftarrow$ empty set;
for $word,\ n \in V_c, C_c$ **do**
    Update(V, word);
    Update(P, word, $w_1 \cdot \frac{n}{sum_c}$);
end
for $word,\ n \in V_d, C_d$ **do**
    Update(V, word);
    Update(P, word, $w_2 \cdot \frac{n}{sum_d}$);
end

Algorithm 2: Word Segmentation

Input: Vocabulary set V, probability set P, URL u
Output: Word token list ws

$ws \leftarrow$ empty list;
re $\leftarrow$ "[a-zA-Z]+|[0-9]+";
for *token* $\in$ *Tokenize(re, u)* **do**
    if *Match(re, token)* **then**
        Append(ws, Viterbi(token, V, P)));
    else
        Append(ws, token);
    end
end
Function *Tokenize(re: regexString, u: URL)*
    str $\leftarrow$ u;
    token_list $\leftarrow$ empty list;
    while *str not empty* **do**
        start, end $\leftarrow$ RegexFind(re, str);
        prefix $\leftarrow$ str[:start];
        substr $\leftarrow$ str[start:end];
        str $\leftarrow$ str[end:];
        Append(token_list, prefix);
        Append(token_list, substr);
    end
    return token_list;
end

URLs collected from Common Crawl as input.



(a) Length of URL (b) Length of Word

Fig. 4. Distribution of URL Length and Word Length

B. Baselines

We consider seven state-of-the-art models as baselines, which can be categorized into heuristic-based, machine learning-based and deep learning-based methods. To build the heuristic-based baselines, 16 handcrafted features are selected from previous works [4], [25], wherein 12 of them are URL-based lexical features while the other 4 are host-based information retrieved from Whois[8]. We then apply three prevalent machine learning algorithms implemented by scikit-learn[9], including Random Forest (RF), Logistic Regression (LR), and Support Vector Machine (SVM), on the 16 handcrafted features to train three classifiers, respectively. As for the machine learning-based ones, we extract bag-of-words features of URLs based on the approach proposed by Kolari et al. [28]. After that, we apply the same three algorithms (i.e., RF, LR, SVM) on the bag-of-words features to build another three classifiers. To compare our approach with a deep learning-based one, we select URLNet [5] as another baseline.

C. Evaluation metrics

Due to the relatively high percentage of legitimate websites compared to the phishing ones, phishing URL detection is essentially a classification task on imbalanced datasets. Considering that AUC is insensitive to the imbalance of datasets, we quantify the performances of both our approach and the baselines using AUC as an evaluation metric. Besides, we evaluate their accuracy, false positive rate, and precision to fully assess the effectiveness of the models.

D. Network Configuration

As seen in Fig. 4, albeit the length of URLs (number of characters) varies from tens to hundreds, most URLs have less than 200 characters. Moreover, it is apparent that the length of most words in URLs is shorter than 32. Thus we set L_w and L_c to 200, and L_{wc} to 32. Besides, the embedding dimension k is set to 128 empirically.

The attention-based hierarchical RNN module contains 2 bidirectional LSTM layers with 128 hidden units. The attention size of BahdanauAttention [29] is set to 256 for both character-level and word-level attention blocks. As for the character-level CNN module, refer to Table III for the detailed configuration.

[8]https://www.whois.net/
[9]https://scikit-learn.org/stable/

TABLE III
CONFIGURATION OF CHARACTER-LEVEL CNN MODULE. K CAN BE 2,3,4,5,6. CH I/O: NUMBER OF CHANNELS OF INPUT/OUTPUT FEATURES. IN/OUT DIM: INPUT/OUTPUT DIMENSION.

Block	Kernel	Stride	Ch I/O	In Dim	Out Dim
Conv_BN_ReLU_k_1	k × 128	1 × 1	1/128	200 × 128	200 × 1
Conv_BN_ReLU_k_2	2 × 1	2 × 1	128/64	200 × 1	100 × 1
Conv_BN_ReLU_k_3	2 × 1	2 × 1	64/32	100 × 1	50 × 1
Conv_BN_ReLU_k_4	2 × 1	2 × 1	32/16	50 × 1	25 × 1

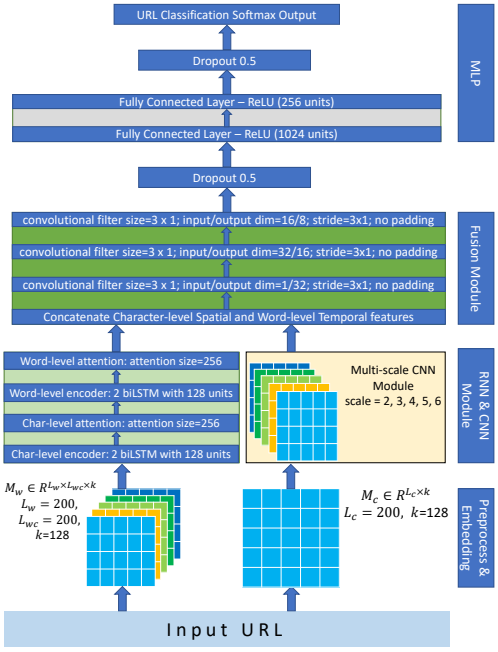


Fig. 5. Network Configuration

For the Adam optimizer, we set the learning rate to 0.0001. The dropout rate is set to 0.5 while the weight decay is 0.001. Empirical evidence shows that when the θ in (5) is set to 0.5, our approach can reach the best result. Other detail about the network configuration can be referred at Fig. 5.

Our approach is implemented based on TensorFlow [30].

E. Results and Discussion

The experimental results can be seen in Table IV. First, we compare the performance of our approach against the baselines. Then we show the effectiveness of our novel objective function. Next, we present the incremental performance gain of each feature extraction module.

Our approach outperforms the baselines across all the evaluation metrics significantly. Generally, classifiers trained on bag-of-words are inclined to have better performances than those trained on handcrafted features, which means that heuristic-based handcrafted features are not always effective since it is easy for phishers to evade such detections. Within the baselines, we observe that while random forest classifiers can reach the best AUC, SVM classifiers trained on bag-of-words get the highest accuracy. This is because SVM relies on the distance between different data for classification while the extracted features are a mixture of numerical and categorical features, which is unsuitable for distance calculation but easy for random forest to handle. Besides, heuristic-based logistic regression classifier has the lowest false positive rate and the highest precision score while it gets the worst AUC, which means that it classifies lots of phishing URLs as legitimate ones. However, URLNet is not superior to the machine learning classifiers trained on bag-of-words features in our experiments since it could not handle the rare words well enough to build effective feature representations of URLs.

We conducted four experiments to show the effectiveness of our objective function. Specifically, we trained PhishingNet(CNN), a simplified model of our approach using only the character-level CNN module to extract feature representations of URLs, with four different objective functions, namely traditional cross-entropy loss, $\mathcal{L}_1$ loss, $\mathcal{L}_2$ loss, and $\mathcal{L}$ loss respectively. As shown in Table IV, PhishingNet(CNN) trained with $\mathcal{L}$ loss reaches the best performances across all the evaluation metrics among these four models, demonstrating that our novel objective function $\mathcal{L}$ (i.e., the combination of our optimized cost-sensitive cross-entropy objective function $\mathcal{L}_1$ and the $\mathcal{L}_2$ objective function) is effective for phishing URL detection problem.

To explore the incremental performance gain of each feature extraction module in our approach, we conducted ablation experiments, whose results can be referred in Table IV. Although PhishingNet(CNN) is able to achieve better AUC and accuracy than the baselines, its false positive rate and precision are worse than some state-of-the-art solutions. PhishingNet(RNN), another simplified model of our approach using only the attention-based hierarchical RNN module to extract feature representations of URLs, fails to extract effective feature representations from URLs due to the complexity of segmented words. As is discussed above, there is an enormous amount of meaningless rare words in URLs, which may reduce the effectiveness of the word-level temporal feature representations of URLs. Even though we have designed a more effective segmentation algorithm, some segmented words may still lack semantic meanings, which may account for the failure of word-based phishing detection. However, the proposed model PhishingNet(CNN+RNN), which utilizes the character-level CNN module and the attention-based hierarchical RNN module to extract character-level spatial feature representations and word-level temporal feature representations of URLs, outperforms all the baselines across all the evaluation metrics, demonstrating the effectiveness of our approach. Therefore, our experimental results show that spatial and temporal feature representations extracted from URLs using our character-level CNN module and attention-based hierarchical RNN module are powerful and can improve the performance and the generalization ability of our approach.

V. CONCLUSION

In this paper, we proposed an effective deep learning-based phishing URL detection approach and showed the superior ef-

TABLE IV
PERFORMANCE ON TEST DATA. THE BEST PERFORMANCES HAVE BEEN BOLDED.

Model	AUC	ACC	FPR	Precision
Random Forest on Handcrafted Features	0.92208	0.96892	0.00711	0.78360
Logistic Regression on Handcrafted Features	0.83802	0.96369	0.00082	0.94594
SVM on Handcrafted Features	0.83235	0.96781	0.00283	0.87793
Random Forest on Bag of Words	0.95515	0.94108	0.05829	0.45856
Logistic Regression on Bag of Words	0.94474	0.93292	0.06667	0.42655
SVM on Bag of Words	0.94726	0.97709	0.00430	0.90592
URLNet	0.94072	0.88447	0.12004	0.25544
PhishingNet(CNN) with cross-entropy loss	0.98815	0.91113	0.09274	0.35800
PhishingNet(CNN) with $\mathcal{L}_1$ loss	0.98473	0.91239	0.09098	0.36049
PhishingNet(CNN) with $\mathcal{L}_2$ loss	0.97476	0.91669	0.08618	0.37180
PhishingNet(CNN) with $\mathcal{L}$ loss	0.98896	0.97656	0.00932	0.79949
PhishingNet(CNN)	0.98896	0.97656	0.00932	0.79949
PhishingNet(RNN)	0.87186	0.65902	0.35128	0.11353
PhishingNet(CNN+RNN)	**0.99269**	**0.97905**	**0.00020**	**0.98958**

fectiveness of our approach with extensive experiments against several state-of-the-art solutions on the large URL dataset we built. As the results suggest, our approach outperforms the state-of-the-art solutions significantly with the help of our novel objective function. Meanwhile, the ablation experiments show that the character-level spatial feature representations and the word-level temporal feature representations extracted from URLs via our character-level CNN and attention-based hierarchical RNN modules can improve the performance and the generalization ability of our approach greatly.

ACKNOWLEDGMENT

This research has been supported by grant 2017YFB0202502 and SQ2017YFGX060171-03 in the National Key R&D of China.

REFERENCES

[1] G. Aaron, "Phishing activity trends report, 2nd quarter 2018." [Online]. Available: http://docs.apwg.org/reports/apwg_trends_report_q2_2018.pdf

[2] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012.

[3] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (sok): A systematic review of software-based web phishing detection," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2797–2819, 2017.

[4] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing urls using recurrent neural networks," in *Electronic Crime Research (eCrime), 2017 APWG Symposium on*. IEEE, 2017, pp. 1–8.

[5] H. Le, Q. Pham, D. Sahoo, and S. C. Hoi, "Urlnet: Learning a url representation with deep learning for malicious url detection," *arXiv preprint arXiv:1802.03162*, 2018.

[6] J. A. Hanley and B. J. McNeil, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases." *Radiology*, vol. 148, no. 3, pp. 839–843, 1983.

[7] Y. Wang, R. Agrawal, and B.-Y. Choi, "Light weight anti-phishing with user whitelisting in a web browser," in *Region 5 Conference, 2008 IEEE*. IEEE, 2008, pp. 1–4.

[8] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *INFOCOM, 2010 Proceedings IEEE*. Citeseer, 2010, pp. 1–5.

[9] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 639–648.

[10] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 2, p. 21, 2011.

[11] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious url detection using machine learning: A survey," *arXiv preprint arXiv:1701.07179*, 2017.

[12] Q. Cui, G.-V. Jourdan, G. V. Bochmann, R. Couturier, and I.-V. Onut, "Tracking phishing attacks over time," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 667–676.

[13] W. Zhang, Y.-X. Ding, Y. Tang, and B. Zhao, "Malicious web page detection based on on-line learning algorithm," in *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, vol. 4. IEEE, 2011, pp. 1914–1919.

[14] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from urls," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.

[15] J. Saxe and K. Berlin, "expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys," *arXiv preprint arXiv:1702.08568*, 2017.

[16] B. Athiwaratkun and J. W. Stokes, "Malware classification with lstm and gru language models and a character-level cnn," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2482–2486.

[17] M. Nguyen, T. Nguyen, and T. H. Nguyen, "A deep learning model with hierarchical lstms and supervised attention for anti-phishing," *arXiv preprint arXiv:1805.01554*, 2018.

[18] J. Saxe, R. Harang, C. Wild, and H. Sanders, "A deep learning approach to fast, format-agnostic detection of malicious web content," *arXiv preprint arXiv:1804.05020*, 2018.

[19] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: Url names say it all," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 191–195.

[20] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[21] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[22] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[23] L. Yan, R. H. Dodier, M. Mozer, and R. H. Wolniewicz, "Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 848–855.

[24] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[25] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1245–1254.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[27] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[28] P. Kolari, T. Finin, A. Joshi *et al.*, "Svms for the blogosphere: Blog identification and splog detection," in *AAAI spring symposium on computational approaches to analysing weblogs*, 2006.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/