Robust URL Phishing Detection Based on Deep Learning

Abdullah Al-Alyan[1*] and Saad Al-Ahmadi[2]
[1] Department of Computer Science, King Saud University
Riyadh 11543, KSA
[e-mail: AbdullahAlyan@yahoo.com]
[2] Department of Computer Science, King Saud University
Riyadh 11543, KSA
[e-mail: salahmadi@ksu.edu.sa]
*Corresponding author: Abdullah Al-Alyan

Received December 27, 2019; revised March 6, 2020; revised May 8, 2020; accepted June 10, 2020; published July 31, 2020

Abstract

Phishing websites can have devastating effects on governmental, financial, and social services, as well as on individual privacy. Currently, many phishing detection solutions are evaluated using small datasets and, thus, are prone to sampling issues, such as representing legitimate websites by only high-ranking websites, which could make their evaluation less relevant in practice. Phishing detection solutions which depend only on the URL are attractive, as they can be used in limited systems, such as with firewalls. In this paper, we present a URL-only phishing detection solution based on a convolutional neural network (CNN) model. The proposed CNN takes the URL as the input, rather than using predetermined features such as URL length. For training and evaluation, we have collected over two million URLs in a massive URL phishing detection (MUPD) dataset. We split MUPD into training, validation and testing datasets. The proposed CNN achieves approximately 96% accuracy on the testing dataset; this accuracy is achieved with URL schemes (such as HTTP and HTTPS) removed from the URL. Our proposed solution achieved better accuracy compared to an existing state-of-the-art URL-only model on a published dataset. Finally, the results of our experiment suggest keeping the CNN up-to-date for better results in practice.

Keywords: Phishing Detection, Machine Learning, Deep Learning, Convolutional Neural Network (CNN), Cyber Security

1. Introduction

Phishing websites attempt to impersonate other websites or entities to convince users to enter their personal information. This makes phishing websites dangerous, especially for amateur users. Various approaches have been proposed in the literature for protecting against phishing, one of which is employing machine learning for the automatic detection of phishing websites. However, many state-of-the-art phishing detection models have been evaluated on small data sets. This can be seen in the related work section, where seven of the eleven models we reviewed were trained and evaluated on data sets which had only up to 3000 instances. In addition, evaluations are less reliable when the dataset is small, especially for classifiers with a high number of features or with complex classification rules [1]. In addition, non-simple phishing detection models which require complex features are harder to adopt depending on the usage scenario. For example in firewalls, which may have a limited storage, restricted connectivity, and need for high throughput. Similarly, it may not be preferable for web browsers to utilize models which introduce unnecessary network delay.

In this paper, we propose a URL phishing detection solution which utilizes a character-level convolutional neural network (CNN) model to classify the URL. The CNN learns from the URL string as a character sequence, in order to avoid predetermined URL features (e.g., the number of dots or the length of the URL). Many of the methods reviewed in the related works have been based on predetermined URL features. On the other hand, with the proposed model, we explore a different direction, which achieves better results while also being more convenient. We believe that other approaches, such as n-gram and bag-of-words models, are less suitable for URL phishing detection, compared to character-level classification, for two reasons: The first reason is that URLs are harder to tokenize, unlike sentences (which are split naturally by spaces). The second reason is that the ordering of words for domain names is very important; for example, domains like login.some.com and some.login.com are totally different domains and may refer to different pages. The advantage of being URL-only means that the model does not depend on any third-party service or network connectivity.

For our proposed solution, we propose a character-level CNN architecture. To train and evaluate the CNN, we have collected and preprocessed a large data set of more than two million unique URL instances. The data set contains over 1 million verified phishing websites from PhishTank and over 1 million legitimate websites sampled from the top 4 million domains. We would like to emphasize that our data set is much larger than the data sets used in the works we review in the related works section, in which the largest data set had 26,052 instances after removal of duplicated hosts. In addition, we removed the URL scheme to avoid making the CNN overly dependent on the scheme. Using a simple decision rule with URL schemes such HTTPS, approximately 74% accuracy can be achieved on the constructed data set. To evaluate the CNN, we split the data set into training, validation, and testing data sets, and the proposed CNN achieved approximately 96% accuracy on the testing data set. The large data set is a prime focus of our paper, as it helps to address sampling issues and overfitting, which may inflate the reported accuracy. For example, a data set that contains only legitimate websites from the top 1 thousand websites is much less representative of legitimate websites, in general, than our data set, which contains legitimate URLs from the top 4 million domains. This is problematic when ranking information is used as a feature for the model, as ranking information alone is enough to get very high accuracy on the less representative data set.

We do not find direct comparisons with reported accuracies very useful, due to differences such as the methodologies and the data sets used. This is especially true when the difference is very small. However, we provide direct comparisons against a state-of-the-art URL-only model, which achieved high accuracy. To provide these comparisons, we evaluate our proposed model using the same data set used to evaluate the state-of-the-art model. In addition, to provide competing benchmarks on our data set, we trained various machine learning models on predetermined URL features which have been commonly used in the literature. Furthermore, to evaluate how our proposed solution may fare in the future, we evaluated our proposed solution with a different data set split. The training data set contained phishing instances from September 2006 to September 2013. The validation data set was from September 2013 to August 2015. The testing data set was from August 2015 to October 2018. This resulted in a drop of 7.5% in the accuracy of the CNN. This drop is significant, considering phishing instances are roughly half of the data set. Thus, keeping the model up to date is recommended for better accuracy in practice. In addition, the results for the other models may suggest that URL phishing instances are not independent and identically distributed.

2. Background

In this section, we briefly describe Neural Networks as our proposed solution utilizes a CNN, a type of neural network. Neural networks are composed of one or more layers, where each layer is linear combination of the previous layer and the first layer is linear combination of the input. To add non-linearity, an activation function is applied to the output of each layer. Neural networks can be succinctly described by the following equation:

$$h^i = g^i\left(W^{(i)T} h^{i-1} + b^i\right) \quad (1)$$

where h^i is the output vector of layer i, g^i is the activation function of layer i, W^i is the weight matrix of layer i, b^i is the bias vector of layer i, and h^0 is the input vector.

Neural networks are capable of approximating any continuous function. By simply specifying the architecture of the neural network, many optimization algorithms can be used to find weight matrices and bias vectors which approximate the training data set. However, for complex tasks, the number of weights is very large, which makes training impractical. A CNN is a specialized type of Neural Network, which keeps the number of weights manageable by using convolution and pooling layers. There exist many character-level CNN architectures in the literature, such as those in [2] and [3]. These CNNs have already achieved good results in various text classification and language modeling tasks.

3. Related Works

We categorize the machine learning solutions surveyed here into URL exclusive—which only utilizes the URL—and non-URL exclusive—which can utilize any information. Our proposed solution mainly competes with the URL exclusive solutions, as it belongs to the same category and has the same usage scenarios.

The main critique we have for most of the works we surveyed is the small sizes of the datasets used, which may decrease confidence in the evaluation of the models as a small data set is usually accompanied with difficulties in sampling. For example, a set of legitimate

websites collected from top 1000 popular websites probably represent popular websites more than legitimate websites. Furthermore, smaller data sets are more prone to overfitting. The main difficulty on collecting phishing websites, which limited the size of the data sets in most works, is that phishing websites are usually short-lived. Even if the website is accessible and gives a response, it may not be the phishing website itself; it could, for example, be a response from the domain provider that the domain is no longer used. Many features require the phishing website to be online, such as the content of the page. Websites like PhishTank record phishing URLs, but do not preserve their content.

3.1 URL Exclusive

Sahingoz et al. [4] proposed a URL-only phishing detection solution which had various predetermined URL features, called NLP features. The NLP features were human-determinable features which could be computed by a program. The NLP features (of which there were 40) included: Raw Word Count, Brand Check for Domain, Average Word Length, and so on. Their data set contained 36,400 legitimate URLs and 37,175 phishing URLs, which they have published. The phishing URLs were mostly collected from PhishTank. The legitimate URLs were found by querying the Yandex Search API using a specific keyword list and taking the highest-ranking pages, which would thus naturally have a low chance of being phishing websites. The authors used 10-fold cross-validation to evaluate 7 different classifiers and three variants of the features: NLP features, word vectors, and hybrids. Out of these 21 model variants, RandomForest-NLP which used RandomForest with only the NLP features had the best accuracy (97.98%). While experimenting on their data set, we found that their data set contained 18,812 repeated instances and, out of the remaining instances, 28,711 instances had a repeated host, which may have inflated the reported accuracy.

On the other hand, Zouina et al. [5] proposed another URL phishing detection solution. Their solution was based on a SVM with Gaussian kernel and targeted mobile phones and other embedded devices, which have less computation power. To achieve the goal of being lightweight, the system used only 6 URL features: URL length, the number of hyphens, the number of dots, the number of numeric characters, the presence of an IP address, and the Hamming distance from the target website. The author experimented with various distance metrics and found that the Hamming distance had the lowest error rate for SVM. Their data set contained 1000 phishing websites and 1000 legitimate websites, where the phishing websites were taken from PhishTank. The legitimate websites were collected from Amazon Alexa's top 500, in addition to websites collected from queries about banking, trading, and commerce on Google. The system achieved 95.80% accuracy using five-fold cross-validation. A limitation of this approach was the assumption that the target website was known. Even if we take the website with minimum distance (excluding the website itself) the accuracy and performance could be affected and need re-evaluation. In addition, the legitimate websites considered were only high-ranking websites, which may not be representative of all legitimate websites.

3.2 Non-URL Exclusive

With a focus on internet banking, Moghimi et al. [6] proposed a rule-based solution for phishing detection. The proposed solution depended only on the URL and the page served. Their work used two novel feature sets, in addition to five features which they chose by analyzing the usage frequency of various features used in the literature. The first novel feature set used the average Levenshtein distance between the domain and the linked resources for each type of resource. The second feature set was the rate of secure access for linked pages for each type of resource. The authors collected data from two sources: Yahoo Directory for

legitimate pages, and PhishTank for phishing pages. The collection process was focused on internet banking. After preprocessing, the collected data set contained 1448 phishing websites and 686 legitimate websites. The data set was split into training and testing data sets in a ratio of 80% to 20%, respectively. The authors first trained a support vector machine classifier with polynomial kernel of degree 3. Later, the authors used a method from the literature to extract approximate, yet simple and explainable rules from the support vector machine classifier. The proposed solution achieved 98.86% accuracy on the testing data set. However, the data set used was small; especially the test data set, which contained only 103 and 73 phishing and legitimate websites, respectively.

Similarly, Montazer et al. [7] proposed a phishing detection system for e-banking. The authors prepared 28 features at the start, using a feature selection algorithm based on rough set theory on 50 instances of banking websites. They determined that six features were the most distinguishing and disregarded the others. The six features were Length of URL, Certificate authority, Distinguished names certificate, Abnormal URL of anchor, Abnormal server form handler (SFH), and Adding a prefix or suffix. The authors used five classes for the websites, with varying degrees from legitimate to phishing. The author used a fuzzy expert system with these features. The proposed solution showed 88% accuracy on the 50 banking websites.

Abdelhamid et al. [8] tackled the problem of website phishing detection using Multi-label Classifier-based Associative Classification, which provided an explainable model. The data set used contained 16 features, which were chosen from the literature based on frequency analysis against the data set. The authors collected data from Yahoo Directory, PhishTank, and Millersmiles, in total collecting over 1350 different websites. The proposed solution achieved roughly 94% accuracy.

Using the data set from [9], Ferreira et al. [10] proposed an ANN model for phishing detection. The data set contained 30 features and 11,050 records. The authors reduced the size of the data set to 3000 for convenience of preprocessing, on which they trained the model on 2000 records from the dataset (where it achieved an accuracy of 87.61%) and tested it on the remaining 1000 records, on which it achieved an accuracy of 98.23%. Although the model achieved high accuracy on the test data, its performance on the training data itself was lacking.

Furthermore, Babagoli et al. [11] used the same data set from [9] to experiment with feature selection methods and meta-heuristic algorithms. The authors utilized wrapper, which they found superior to decision tree-based feature selection, to select 20 features. The models used were non-linear regression based on a modified harmony search and SVM. The models achieved 92.80% and 91.83%, respectively, using 10-fold cross-validation.

Adebowale et al. [12] proposed a phishing detection system which used an Adaptive Neuro-Fuzzy Inference System algorithm for classification. The proposed solution considered the features of Search index, Security & encryption, URL, Domain identity, JavaScript, frame, and images. The authors used both information gain and chi-square methods for feature selection. The authors combined two existing data sets of sizes 8355 and 2500. Both had the same 30 attributes. The authors added an additional 2145 websites from PhishTank, making the overall data set 4898 phishing websites, 1945 suspicious websites, and 6157 legitimate websites. Their approach showed 98.3% accuracy.

Li et al. [13] proposed a phishing detection system based on minimum enclosing ball SVM classification, which has a shorter training time compared to regular SVM. The system had 12 features, which were extracted from the DOM tree of the web page. The authors collected 400 phishing websites and 400 legitimate websites from PhishTank and from well-known classified websites. Their proposed solution achieved 0.963 F-measure on a testing sample of

100 phishing websites and 100 legitimate websites. The main limitation was that the test sample was very small.

Abutair et al. [14] proposed a weighted KNN model, where the weights and features selected were based on information gain and genetic algorithms. Their work was evaluated on two collected data sets of sizes 500 and 750. The features used were mostly extracted from the URLs, in addition to various features related to ranking and internet presence. The best models achieved 95.81% accuracy on the smaller data set and 96.26% on the larger data set.

Finally, Yi et al. [15] proposed a deep learning phishing detection solution. Their proposed solution is very different from our proposed solution as it depend on complex predetermined features. The proposed solution is based on a deep belief network with SVM on the last layer. The proposed solution depends on three binary features which are based on URL and domain age. Additionally, the proposed solution depends on five interaction-based features which are the frequency of visits from user, the usage of a well-known browser, existence of cookies and two other features which were based on the graph structure of traffic flow. The proposed solution was trained and evaluated based on real IP flows from an Internet service provider. It was trained on a data set based on 40 minutes of traffic and was evaluated a data set based on 24 hours of traffic. The dataset was labeled by some blacklist. The training data included 36,729 unique URLs, while the testing data set included 1,982,005 unique URLs. The proposed solution achieved 89.6% accuracy and 89.2% recall.

Table 1 summarizes all the models discussed. Most of the models achieved high accuracies; however, we cannot use these accuracies for accurate benchmarking, due to the differences in the datasets and evaluations, especially when the differences were small.

Table 1. Summary of Related Works

Paper	Model	Dataset size	Accuracy
Sahingoz et al. [4]	Random Forest	73,575	97.98%
Zouina et al. [5]	SVM with Gaussian kernel	2000	95.80%
Moghimi et al. [6]	Extracted rules from third-degree polynomial SVM	2134	98.86%
Montazer et al. [7]	Fuzzy expert system	50	88%
Abdelhamid et al. [8]	Multi-label Classifier-based Associative Classification	1350	94%
Ferreira et al. [10]	ANN	3000	98.23%
Babagoli et al. [11]	Non-linear regression based on modified harmony search	11,050	92.80%
Adebowale et al. [12]	Adaptive neuro fuzzy inference system	13,000	98.3%
Li et al. [13]	Minimum enclosing ball SVM	800	NA
Abutair et al. [14]	Weighed KNN based on genetic algorithms and information gain	750	96.26%
Yi et al. [15]	Deep belief network with SVM on last layer	2,018,734	89.6%

4. Methodology

In this section, we discuss the used data sets, the preprocessing steps performed, our proposed CNN, the competing models, the experimental setup, and the relevant statistical measures. **Fig. 1** gives an overview of our proposed solution.

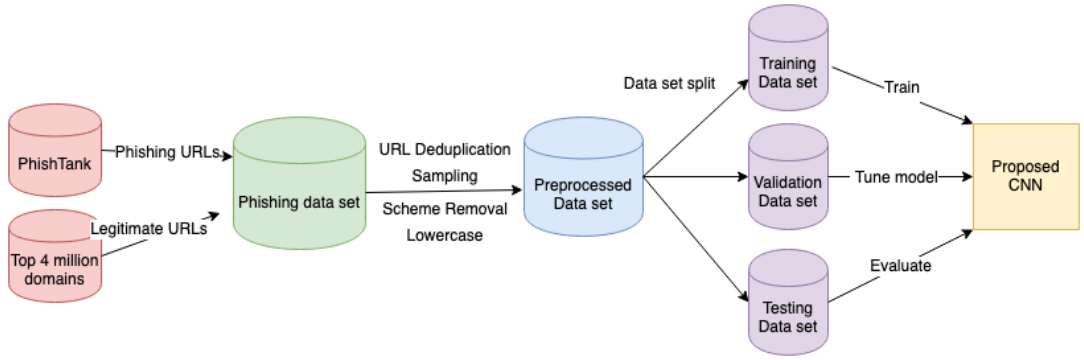

Fig. 1. Overview of the Proposed Solution.

4.1 Data sets

In our experiments, we use two data sets: the large data set we collected, comprised of 2,220,853 legitimate URLs and 2,353,933 phishing URLs. As shorthand, in this paper we will refer to this large URL phishing detection data set as the MUPD data set. We will describe the collection process of the MUPD data set in this section. In addition, we also use the data set published by Sahingoz et al. [4], which we will refer to as the Sahingoz data set.

The MUPD data set had a different source for legitimate and phishing URLs. The source for phishing website URLs was PhishTank, which was similarly used by most of the works we reviewed in the related works section. We only considered phishing websites which were verified as phishing on PhishTank. PhishTank lets users publish and verify phishing websites. PhishTank defines phishing as "a fraudulent attempt to get you to provide personal information, including, but not limited to, account information". For practical purposes, we defined the phishing websites in the MUPD data set as those websites that were verified on PhishTank. This practical definition may sometimes conflict with what is usually regarded as a phishing website, because the users who submit and verify websites as phishing may make mistakes or have different ideas of what is considered phishing. In short, we declare that all considerations about how accurate phishing URLs in the MUPD are outside the scope of this paper. Finally, we note that PhishTank included the reported date of each phishing URL, which we make use of in one of our experiments. The phishing URLs we retrieved were dated from September 29th, 2006 to October 30th, 2018. The main critique we have for most of the works we surveyed is the small sizes of the datasets used, which may decrease confidence in the evaluation of the models as a small data set is usually accompanied with difficulties in sampling. For example, a set of legitimate websites collected from top 1000 popular websites probably represent popular websites more than legitimate websites. Furthermore, smaller data sets are more prone to overfitting. The main difficulty on collecting phishing websites, which limited the size of the data sets in most works, is that phishing websites are usually short-lived. Even if the website is accessible and gives a response, it may not be the phishing website itself; it could, for example, be a response from the domain provider that the domain is no longer used. Many features require the phishing website to be online, such as the content of the page. Websites like PhishTank record phishing URLs, but do not preserve their content.

The source for the legitimate websites was the top 10 million domains list from DomCop [16]. This ranking is based on data that had been collected over the 7 years before July 2017. We assumed that the running websites on the top 4 million domains from this list were legitimate. Similar assumptions have been used in many of the related works discussed above.

We believe that this assumption is reasonably accurate for two reasons: The first is that we observed that most phishing websites are short-lived, whereas this data was over two years old. The second reason is that gaining and maintaining page rank is not easy, more so for phishing websites which would be black-listed and reported, for example, by web browsers. Similarly, to phishing websites any considerations of how accurate the legitimate URLs in the MUPD data set are outside the scope of this paper. To collect the legitimate URLs, we wrote a program to retrieve the index page (if it existed) for each of the top 4 million domains and, from the index page, retrieved a random internal URL. This was done to avoid using only index URLs, which would make classification simple and useless as a simple rule to classify index pages as legitimate websites would achieve very high accuracy.

4.2 Preprocessing

We performed the following preprocessing steps to generate the data sets we published: sampling to ensure a balanced data set, data deduplication, and splitting the data set into training, validation, and testing data sets. We performed two additional preprocessing steps in memory: URL scheme removal and conversion of URLs to ASCII lower case.

Due to the nature of the collection process, the collected data sets usually contained many repeated URLs or different URLs from the same host. For example, many pages from the same phishing website were frequently reported as phishing pages. Similarly, our collection process using the top domains may have resulted in repeated hosts due to various reasons, such as http redirects. This was not exclusive to our collection process, as the Sahingoz data set also contained repetitions. When we performed data deduplication, we removed repeated URLs and URLs with repeated hosts. The goal of the deduplication was to make the evaluation fairer and to prevent models from memorizing the host.

Having a balanced data set is usually preferable in binary classification problems, especially when the accuracy metric is used. Although the MUPD data set was balanced before deduplication, when the deduplication was performed, the phishing URLs (which were more commonly repeated in the data set) represent roughly only one third of the new data set. To solve this problem, we used a random sample of 1,200,000 legitimate URLs. With this sample, we were able to obtain a balanced data set of1,167,201 phishing URLs and 1,140,599 legitimate URLs after deduplication. For the Sahingoz data set, we ended up with 11,696 phishing URLs and 14,356 legitimate URLs after deduplication, as compared to the original data set which had 36,400 legitimate URLs and 37,175 phishing URLs. **Table 2** summarize the sizes of data sets used in our experiments.

Table 2. Sizes of data sets used in our experiments.

	Legitmate	Phishing
MUPD data set	1,140,599	1,167,201
Sahingoz data set	11,696	14,356
Sahingoz data set (No Preprocesing)	36,400	37,175

Finally, we split each data set into training, validation, and testing data sets in the following proportions: 0.6, 0.2, and 0.2, respectively. We performed the data set split randomly. In addition, for the MUPD data set, we also performed a split based on date, where older phishing URLs were kept in the training data set and newer phishing URLs were kept in the testing data set, with the validation data set being in between. Legitimate instances were always split randomly, because we could not associate time with them (unlike the phishing instances, which had a report date). In the date split for the MUPD data set, the training data set contained

phishing instances from September 2006 to September 2013, the validation data set was from September 2013 to August 2015, and the testing dataset was from August 2015 to October 2018. Interested authors can contact us for the data sets. We encourage the usage of all these data sets to benchmark against our proposed solution.

Depending on the URL scheme, the model may be less robust; so, for our preprocessing, we additionally removed the scheme for all URLs. To illustrate how the scheme may lead to a less robust model, **Table 3** shows the number of occurrences of HTTP and HTTPS for the phishing and legitimate URLs in the MUPD data set. Based on the data in this table, simply predicting legitimacy if the URL scheme was HTTPS and phishing otherwise, an accuracy of 73.98% can be achieved. Such a model is not very robust, because the scheme distribution will probably not stay the same; especially as the use of HTTP becomes associated with phishing or insecure websites. In addition, the use of HTTPS has become easier. Finally, to reduce the size of alphabet needed for our proposed CNN, we converted every URL to ASCII lower case.

Table 3. HTTP and HTTPS occurrences in the MUPD data set.

	HTTP	HTTPS
Legitimate	563,247	577,352
Phishing	1,129,904	37,297

4.3 Proposed CNN

In this section, we detail our proposed phishing URL CNN, which we will refer as PUCNN throughout the rest of the paper. We designed the PUCNN architecture based on a few initial experiments on the validation data set. Many of the architectures we experimented with also achieved similar accuracies on the validation data set, and some of the more complex architectures achieved a slightly better accuracy. We opted for our chosen architecture due its simplicity and high accuracy. **Fig. 2** shows the PUCNN architecture. As can be seen from the embedding layer, we limited the input length to 256 letters, where additional letters (if any) are not input to the CNN. We believe that 256 letters would be enough, as most URLs are small and the start of the URL is usually enough, as it contains the domain. We chose an alphabet of 69 letters, including the English lower-case alphabet, numbers, and various other characters.

In the embedding layer, each character from the alphabet is converted into a vector of the embedding size. We chose the embedding size as 128. The embedding layer was the input of the one-dimensional convolutional layer with a tanh activation function and kernel width of 10. The output of the one-dimensional convolutional layer was then used as the input to a global max pooling layer. The output of the pooling layer was used as the input to two fully connected neural network layers with 128 nodes each with a SELU (Scaled Exponential Linear Unit) activation function. The weights of the fully connected layers are initialized using a Lecun Normal Distribution. Finally, the output layers used a Softmax activation function with two output nodes. Similarly, a Sigmoid activation function with one output node may be used, because the problem was binary classification. We used categorical cross-entropy as the loss function and used the Adam optimizer. We set the number of epochs to 25 or 100 depending on the data set size. After each epoch, the model was evaluated on the validation data set. The model with the highest validation accuracy was chosen. **Table 4** summarize the parameters of PUCNN.

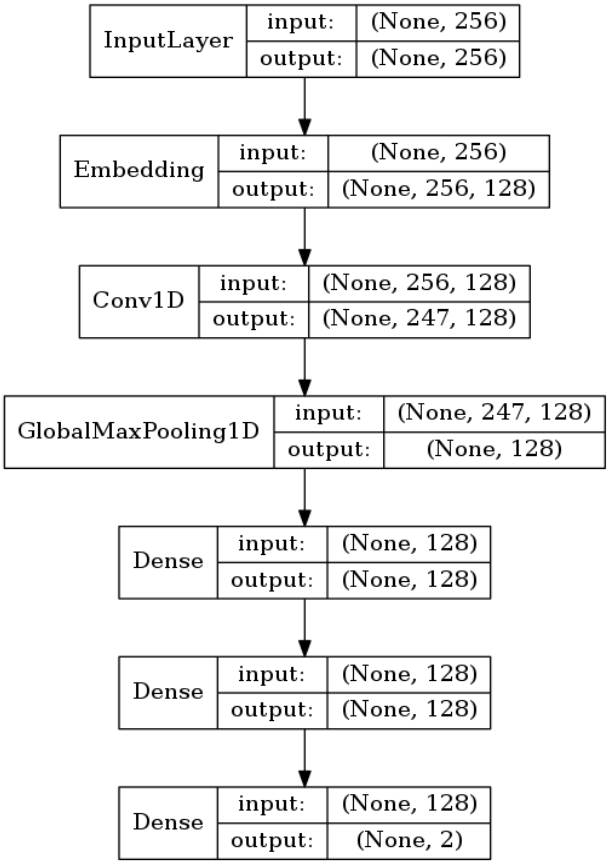

Fig. 2. The Proposed PUCNN Architecture.

Table 4. PUCNN Parameters

Parameter	Value
Input length limit	256
Alphabet	English lower-case alphabet, numbers, and various other characters
Alphabet Size	69
Character embedding size	128
Convolutional layer activation function	tanh
Convolutional layer kernel width	10
Optimization Algorithm	Adam optimizer
Loss function	Categorical cross-entropy
Dense layers weights initialization	Lecun Normal Distribution
Dense Layers activation function	SELU
Output layer activation function	Softmax with two output nodes
Number of epochs	25/100
Model selection criteria	Validation accuracy

4.4 Competing Models

We used the solution by Sahingoz et al [4]—RandomForest-NLP—as our main contender, due

to its various similarities to our proposed solution. First, it utilizes URLs only, making its usage scenarios similar. In addition, it was also trained and tested on the relatively large Sahingoz data set, which increases the confidence of its performance. The Sahingoz data set has been published, so we could train and evaluate our model on it. Furthermore, RandomForest-NLP provides a good contrast for our solution, as it was based on complex predetermined features. Finally, the solution is recent and has achieved high accuracy.

Although RandomForest-NLP was our main contender, we also wanted to see how models based on simple and common URL features would fare. We chose the following features: whether the host is an IP, number of dots, number of hyphens, number of numbers, length, and whether the URL contains an @ symbol. We chose these features because they were common throughout the related works and were simple to replicate.

We performed the experiments with these features on the following models: SVM with linear kernel, SVM with Gaussian kernel, SVM with third-degree polynomial kernel, RandomForest, J48, KNN with K=1, and KNN with K=5. For training the SVM and KNN models on the MUPD dataset, we had to use random sampling to remove half of the data set, as the training time was too long with the full amount. With only these simple features, it may be expected for the models to have fared worse than the CNN; yet, the result of these models is still useful, as it is indicative of the performance of models that use such simple URL features. In fact, many studies in the literature have used models which depend on such simple URL features, in addition to other non-URL features (e.g., ranking).

4.5 Experiments setup

We performed four experiments: The first two experiments were on our MUPD data set and the latter two were on the Sahingoz data set (i.e., the data set used to train and evaluate RandomForest-NLP in [4]). We expected PUCNN to train better and be evaluated more accurately on our MUPD dataset which was much larger; however, with the Sahingoz data set, we were able to perform direct comparisons with the accuracy of RandomForest-NLP. The fourth experiment was the only experiment without data preprocessing, as we needed to be able to perform comparisons with the accuracy of RandomForest-NLP, which was evaluated without our data preprocessing. The difference between the third and fourth experiments was, thus, useful in estimating the effects of the preprocessing in our evaluation. In every experiment—except for experiment 2—the data set was split randomly. In experiment 2, we performed the data set split by date to find approximately how PUCNN would fare, had it been trained once and then used for three years without updating. **Table 5** summarizes the setup of our experiments.

Table 5. Setup of our Experiments.

Ex #	Data set	Epochs	Preprocessing	Split
1	MUPD Data set	25	Yes	Random
2	MUPD Data set	25	Yes	Date
3	Sahingoz Data set [4]	100	Yes	Random
4	Sahingoz Data set [4]	100	No	Random

For the reproducibility of the experiments, we used seeded RNGs. However, we had to use GPUs to train PUCNN in a reasonable time. GPUs, due to their parallel nature, make the results not entirely reproducible and small differences may be perceived. We also note that we could not use 10-fold cross-validation for evaluation, as in the evaluation of RandomForest-NLP, because it was computationally expensive and is not practical for training deep learning models.

4.6 Statistical Measures

Four statistical measures popularly used to determine the accuracy of classification models are precision, sensitivity, F-measure, and accuracy. They are calculated as follows:

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \quad (4)$$

$$F-measure = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (5)$$

where TP, TN, FP, and FN are the occurrence number of the model prediction of true positives, true negatives, false positive, and false negatives, respectively.

We note that a high precision is preferable in situations where false positives are not preferred while a high recall is preferable when false negatives are not preferred. In the case of website phishing detection, high precision means lower number of legitimate websites classified as phishing websites while high recall means lower number of phishing websites which were classified as legitimate. Both of precision and recall are important depending on the usage scenario. For example, it may be preferable to have high precision on personal devices, while in the other hand for some firewalls it may be preferable to have high recall. F-measure is the harmonic mean of precision and recall.

In this work, we mainly used the accuracy measure for comparisons, because it is the common denominator in the related works reviewed; as it was not provided in only [13], which provided F-measure, precision, and recall but did not provide accuracy. However, we also provide the other statistical measures for PUCNN.

5. Results and Discussion

In these sections, we first show benchmarks between accuracies of PUCNN and RandomForest-NLP, the URL-only model we discussed in the related works. Then, we show the accuracy of PUCNN in our MUPD data set and how the type of split affected the accuracy. After that, we also show how the date split affected the accuracy of the other competing models in both the validation and testing data sets. Next, we show the other statistical measures for PUCNN. Finally, we show the accuracies of all the models we trained and evaluated in all the experiments.

Fig. 3 shows the accuracies of PUCNN and RandomForest-NLP on the same data set. The figure provides direct comparisons, as PUCNN was trained and evaluated on the Sahingoz dataset, which was also used for training and evaluating RandomForest-NLP. The main difference in the evaluations was that RandomForest-NLP was evaluated using 10-fold cross-validation, whereas we had to split the data set into training, validation, and testing data sets due to the long training time of PUCNN. PUCNN with data preprocessing had a slightly lower accuracy. This may partly be explained by the number of repetitions being quite high in

the Sahingoz data set, in addition to the removal of URL schemes. We do not know how much the data preprocessing would have affected the accuracy of RandomForest-NLP, but the proposed PUCNN showed a 4.58% decrease in accuracy. However, without the data preprocessing, PUCNN outperformed RandomForest-NLP slightly.

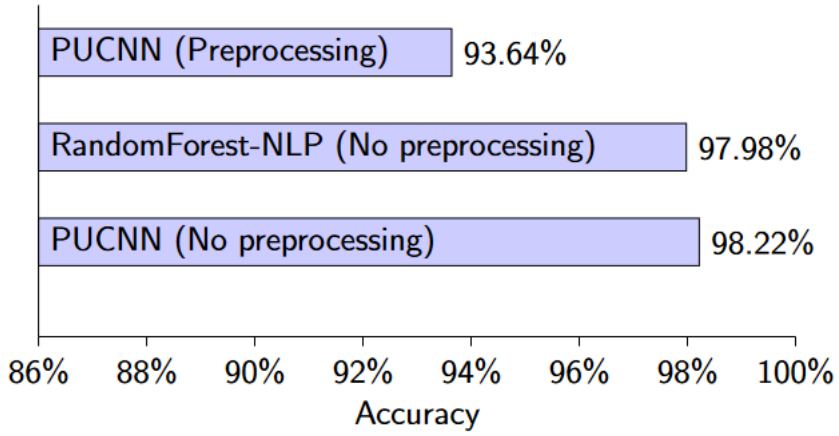

Fig. 3. Direct Comparison of the proposed method with RandomForest-NLP on the Same Data set.

Fig. 4 shows the accuracies of the PUCNN on our MUPD data set with the two different types of data set split. In the random data set split, PUCNN achieved an accuracy of approximately 96%, whereas it achieved approximately 94% accuracy on the smaller Sahingoz data set. This increase could be because the MUPD data set was larger and more representative, improving the models and/or their evaluations. In addition, other differences between the data sets could have had an effect.

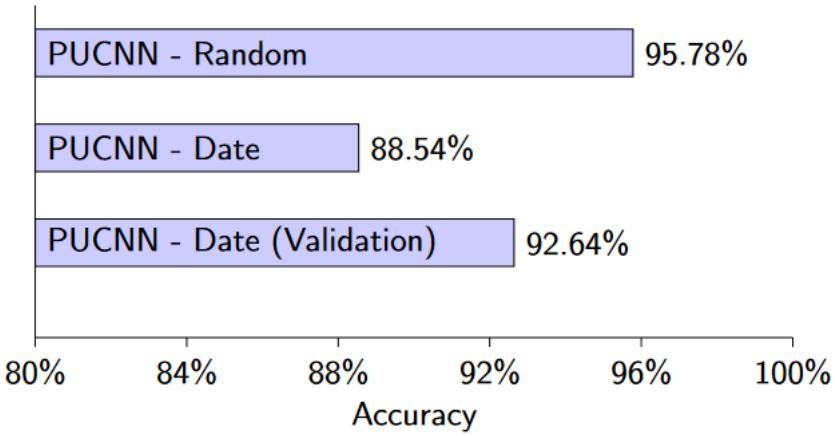

Fig. 4. Accuracy of the proposed PUCNN, based on Type of Split.

As can also be seen from **Fig. 4**, the accuracy of PUCNN decreased by 7.5% when using the date split, in comparison to the random split. In the date split, the training data set contained older phishing instances and the testing data set contained newer phishing instances. This difference was large, especially when considering that phishing URLs represented approximately half of the dataset. These results may indicate that phishing URLs are not independent and identically distributed. and that models should be restrained and improved

more frequently for better results in practice. The models had better accuracies in the validation data set, which was in the middle of training and testing data sets in term of the age of the phishing instances. However, for PUCNN, the model was chosen based on the accuracy of the validation data set, which could be a partial cause of this increase.

Fig. 5 shows how the validation data set also had better accuracies for the other models. We did not include all the models, for sake of brevity. These results are more interesting for the non-CNN models, as the validation data sets were not used in the choice of model at all. This may be an indication that, with time, the characteristics of phishing URLs change; however, further study is necessary in order to draw more definite conclusions.

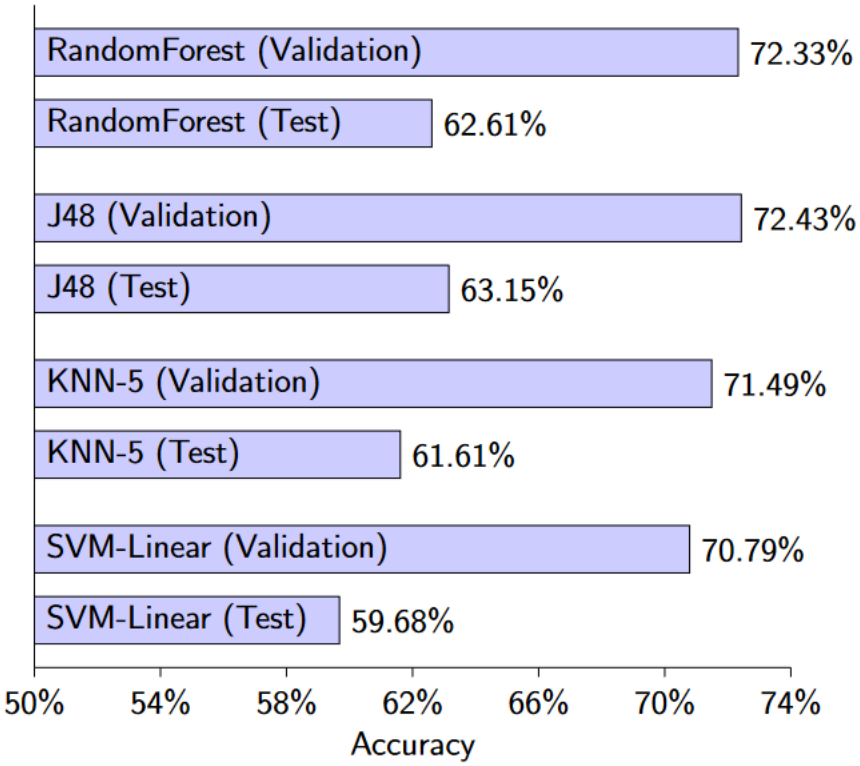

Fig. 5. Validation and Test Data set Accuracies using the Date Split.

In **Fig. 6** we show all the statistical measures for PUCNN. The results are from experiment 1 where we used random sampling to split the MUPD data set. As can be seen from the figure PUCNN showed high results in all the statistical measures. The results are very close to each other; however, we note that that the precision is slightly higher than recall.

The accuracies of every experiment on the testing data set are shown in **Table 6**. PUCNN showed good results in all experiments; although it did not perform as well in experiment 2, where it was trained on phishing instances from 2006–2013 and tested on phishing instances from 2015–2018. Other models had much lower accuracy overall, which may be natural due to the number and simplicity of the features extracted. However, we have seen these simple features used in many of the works we reviewed as URL features, in addition to non-URL features (e.g., ranking). Although the models performed well due to the different types of features they used, by using only these simple URL features they did not utilize the information provided by the URL as well as our model. This could be a key to improving non-URL only models, which we would like to investigate in a future work.

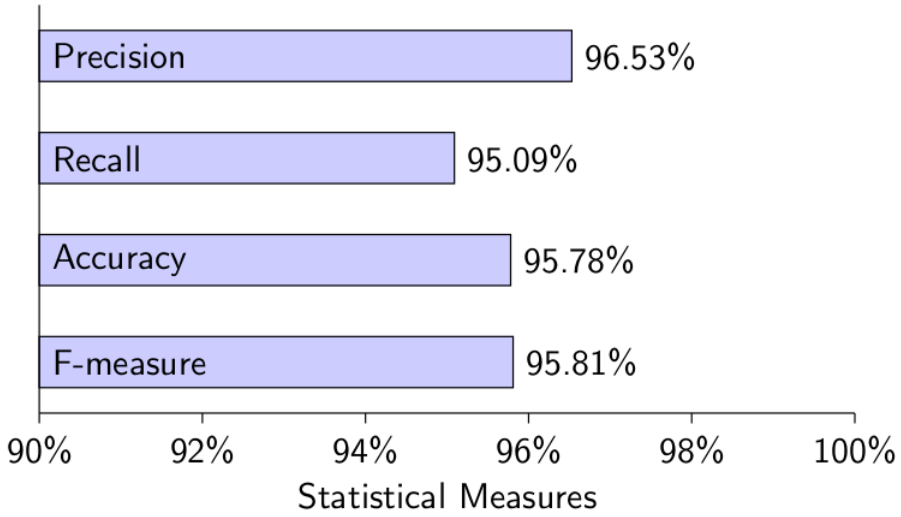

Fig. 6. Results of PUCNN.

Table 6. Overall Accuracies.

Model / Ex#	1	2	3	4
PUCNN	95.78%	88.54%	93.65%	98.22%
SVM-Linear	73.77%	59.68%	63.65%	66.08%
SVM-RBF	71.74%	57.54%	65.86%	65.93%
SVM-Polynomial	50.56%	50.58%	54.25%	52.24%
J48	76.98%	63.15%	73.13%	74.49%
RandomForest	76.62%	62.61%	71.41%	75.11%
KNN-1	71.79%	60.19%	70.04%	74.83%
KNN-5	73.82%	61.61%	71.89%	74.19%

6. Conclusion

In this work, we proposed a robust phishing detection solution which utilizes a character-level CNN on URL data. In contrast to having a set of predetermined URL features, such as the number of dots and the length of the URL, the URL itself was used as the input to a character-level CNN which handles it as a sequence of characters. To train and test our proposed solution, we collected over 2 million URLs, which we split into training, validation, and testing data sets. We proposed a CNN architecture which achieved approximately 96% accuracy on the testing data set. Our proposed CNN achieved this accuracy with the URL scheme removed, meaning that our model did not depend on URL schemes such as HTTPS. Additionally, the proposed CNN outperformed a state-of-the-art URL-only model. In addition, the proposed solution also outperformed various machine learning models based on commonly used simple URL features on the collected data set.

To further evaluate our proposed solution, we additionally performed a data set split of the phishing instances based on the date; that is, the training data set took phishing instances from 2006–2013, the validation data set took phishing instances from 2013–2015, and the testing data set contained phishing instances from 2015–2018. This resulted in a 7.5% decrease of accuracy. Although the accuracy was still good, the decrease was still significant; considering

that the data split affected only the phishing URLs, which were approximately half of the data set. The results of our proposed CNN and the results of the other models suggested that the phishing data we have collected may not be independent and identically distributed and that models may need to be retrained or improved continuously with new data for better results in practice.

References

[1] S. Raudys and A. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, 1991. Article (CrossRef Link)

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

[3] Kim, Yoon, Y. Jernite, D. Sontag, and A. Rush. "Character-aware neural language models," in *Proc. of Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[4] O. 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. Article (CrossRef Link)

[5] M. Zouina and B. Outtaj, "A novel lightweight URL phishing detection system using SVM and similarity index," *Human-centric Computing and Information Sciences*, vol. 7, no. 1, 2017. Article (CrossRef Link)

[6] M. Moghimi and A. Varjani, "New rule-based phishing detection method," *Expert Systems with Applications*, vol. 53, pp. 231-242, 2016. Article (CrossRef Link)

[7] G. Montazer and S. ArabYarmohammadi, "Detection of phishing attacks in Iranian e-banking using a fuzzy–rough hybrid system," *Applied Soft Computing*, vol. 35, pp. 482-492, 2015. Article (CrossRef Link)

[8] N. Abdelhamid, A. Ayesh and F. Thabtah, "Phishing detection based Associative Classification data mining," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948-5959, 2014. Article (CrossRef Link)

[9] R. Mohammad, L. McCluskey and F. Thabtah, "Intelligent rule-based phishing websites classification," *IET Information Security*, vol. 8, no. 3, pp. 153-160, 2014. Article (CrossRef Link)

[10] R. Ferreira et al., "Artificial Neural Network for Websites Classification with Phishing Characteristics," *Social Networking*, vol. 7, no. 2, pp. 97-109, 2018. Article (CrossRef Link)

[11] M. Babagoli, M. Aghababa and V. Solouk, "Heuristic nonlinear regression strategy for detecting phishing websites," *Soft Computing*, vol. 23, no. 12, pp. 4315-4327, 2018. Article (CrossRef Link)

[12] M. Adebowale, K. Lwin, E. Sánchez and M. Hossain, "Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text," *Expert Systems with Applications*, vol. 115, pp. 300-313, 2019. Article (CrossRef Link)

[13] Y. Li, L. Yang and J. Ding, "A minimum enclosing ball-based support vector machine approach for detection of phishing websites," *Optik*, vol. 127, no. 1, pp. 345-351, 2016. Article (CrossRef Link)

[14] H. Abutair, A. Belghith and S. AlAhmadi, "CBR-PDS: a case-based reasoning phishing detection system," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 7, pp. 2593-2606, 2019. Article (CrossRef Link)

[15] P. Yi, Y. Guan, F. Zou, Y. Yao, W. Wang and T. Zhu, "Web Phishing Detection Using a Deep Learning Framework," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1-9, 2018. Article (CrossRef Link)

[16] "Download list of top 10 million domains based on Open data from Common Crawl & Common Search," *Domcop.com*, 2020. [Online]. Available: https://www.domcop.com/top-10-million-domains. [Accessed: 28- Jun- 2019].

Abdullah Al-Alyan was born in Riyadh, KSA. He received his B.S. degree in computer science from King Saud University, Riyadh. He is currently working on his M.S. degree from the same university. He has already finished his coursework and is working on his thesis which is about phishing detection.

Saad Al-Ahmadi is currently an Assistant Professor of Computer Science with the College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He has published many papers in many journals and conferences. His research interests include cyber security, computer networks, mobile ad hoc networks, and sensors networks.