IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 2, 2021, accepted December 7, 2021, date of publication December 23, 2021, date of current version January 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3137636

A Deep Learning-Based Framework for Phishing Website Detection

LIZHEN TANG AND QUSAY H. MAHMOUD, (Senior Member, IEEE)
Department of Electrical, Computer, and Software Engineering, Ontario Tech University, Oshawa, ON L1G 0C5, Canada

Corresponding author: Lizhen Tang (lizhen.tang@ontariotechu.net)

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

ABSTRACT Phishing attackers spread phishing links through e-mail, text messages, and social media platforms. They use social engineering skills to trick users into visiting phishing websites and entering crucial personal information. In the end, the stolen personal information is used to defraud the trust of regular websites or financial institutions to obtain illegal benefits. With the development and applications of machine learning technology, many machine learning-based solutions for detecting phishing have been proposed. Some solutions are based on the features extracted by rules, and some of the features need to rely on third-party services, which will cause instability and time-consuming issues in the prediction service. In this paper, we propose a deep learning-based framework for detecting phishing websites. We have implemented the framework as a browser plug-in capable of determining whether there is a phishing risk in real-time when the user visits a web page and gives a warning message. The real-time prediction service combines multiple strategies to improve accuracy, reduce false alarm rates, and reduce calculation time, including whitelist filtering, blacklist interception, and machine learning (ML) prediction. In the ML prediction module, we compared multiple machine learning models using several datasets. From the experimental results, the RNN-GRU model obtained the highest accuracy of 99.18%, demonstrating the feasibility of the proposed solution.

INDEX TERMS Phishing detection, machine learning, deep learning, RNN-GRU, web browser extension.

I. INTRODUCTION

Internet services have brought tremendous changes to people's lives. Most online services manage users through a membership system, and individual users need to register and log in to obtain these personalized services. Therefore, people need to provide personal information when enjoying these convenient and efficient services. In a secure network environment, the transmission and storage of information are protected by network security technology. However, there are many cybercriminals who use various methods to attack and steal personal information.

Phishing is one of the cyberattack methods that simulate a regular website to trick users into providing personal information. Since the emergence of phishing attacks more than ten years ago, network security experts have been using technical methods to intercept attacks. Attack technology and anti-attack technology are constantly changing and improving. Unfortunately, there is no effective technology that can completely prevent phishing attacks. From the network security reports in recent years, we can see that the economic losses caused by phishing attacks are huge [2]. According to the phishing activity report from APWG, there are more than 100,000 phishing links every month, and it has been a growing trend in the past year [3]. The 2020 annual report from the Internet crime complaint center showed that the economic loss caused by phishing attacks was over $54 million [4].

Commonly used means of spreading phishing links are e-mails, text messages, and social media platforms. The content of the copy edited by the attacker through social engineering means that the user is very eager to click on the phishing link after receiving the information. Therefore, when the phishing link is accessed in the browser, detecting the risk through network security technology, and alerting the user is a very effective anti-attack technology to prevent the user from leaking personal information.

The core of traditional methods of detecting phishing URLs is based on rules. The generation of these rules can be summarized as parsing features from URL and web-page source code and comparing the feature value with

The associate editor coordinating the review of this manuscript and approving it for publication was Wei-Yen Hsu.

an empirical threshold. It can be seen from the academic research report that the number of effective rules is within 100 [5], [6]. Cybercriminals can also develop new attack strategies based on these rules. The rules are interpretable, and the logic of the rules is limited, so the detection methods based on the rules can be easily cracked and used by attackers. For example, the feature of a URL's schema is HTTPS, which is used in many research papers and obtained high importance. However, the APWG report showed an average of 83 percent of phishing websites used HTTPS schema in the first quarter of 2021 [3].

With the rapid development of machine learning, there are more and more applications in the field of cybersecurity. Some scholars and experts have proposed solutions for detecting phishing links based on machine learning, and many academic journal articles show that machine learning-based solutions have achieved high accuracy [7]–[10]. However, in the application scenario of a real-time environment, there are still many challenges. For instance, the real-time system requires the response time of the predictive service to be on the order of milliseconds; a high false-positive rate will affect user experience and user trust.

In this paper, we propose a deep learning-based framework to detect phishing links in a real-time web browsing environment. We developed a browser plug-in to receive client information, call the background prediction service, and show the prediction results to users. When the URL of the current tab of the browser is predicted to be a phishing link, the current page will receive an obvious warning prompt. The prediction result is obtained by the core prediction service calling a trained machine learning model. We introduced multiple models with multiple data sets for comparison and backup. It is concluded from the experimental results that the RNN-GRU model obtains the highest accuracy rate of 99.18, which is better than SVM, Logistic Regression, Random Forest. The contributions of this paper are:

1) A deep learning-based framework for detecting phishing URLs. We trained and tested the models using seven custom datasets generated from four existing data sources, and we achieved the highest accuracy of 99.18% with the RNN-GRU model.
2) A prototype implementation of the proposed framework as a Chrome browser extension.

We organized the rest of the paper as follows: Section II summarizes the related work focusing on deep learning models and real-time frameworks. Section III presents the design and architecture of the proposed framework. Section IV discusses the prototype implementation, including some open-source frameworks, services, and tools that we have utilized. Experimental results and analysis are reported in section V. Finally, Section VI concludes the paper and offers ideas for future work.

II. RELATED WORK

Phishing attacks represent a serious problem, and the technology for detecting and intercepting phishing attacks is constantly evolving. It is the most accurate and fast way to filter good URLs through the whitelist and block phishing URLs through the blacklist. However, the list method cannot detect new phishing links, and because of the low cost of creating a phishing URL, the attacker does not rely on using the same phishing link multiple times.

Many research reports based on machine learning have been published, and high accuracy results have been obtained in experiments. However, in the actual network environment, there are still many victims of phishing attacks every year, causing economic losses. There is still a certain gap between the experimental data results and the real network security solutions. Therefore, it is very important to study anti-phishing solutions in a real-time environment.

We divide the related work into two parts: (1) deep learning-based methods for detecting phishing websites (2) frameworks with prototype implementations.

A. DEEP LEARNING-BASED METHODS

In this part, we reviewed some state-of-the-art deep learning-based solutions for phishing websites detection.

Bu and Cho [11] proposed a deep autoencoder model to detect zero-day phishing attacks and obtained 97.34% accuracy. They extracted character-level features from URL strings and executed experiments on three different datasets collected from Phish Storm [2], ISCX-URL-2016 [12], and Phish Tank [13]. They used receiver-operating characteristic curve analysis and N-fold cross-validation to evaluate the experimental results. Comparing the root mean square error (RMSE) in the reconstruction phase between legitimate URLs and phishing URLs, they found the RMSE increased significantly for the phishing URL.

Somesha *et al.* [14] introduced deep learning models for detecting phishing websites only using ten features extracted from HTML and a third-party service. They compared three deep learning models and calculated 18 features' weights. The experimental results demonstrated that the Long Short-Term Memory (LSTM) model achieved the highest accuracy of 99.57%. However, they only used one published dataset with 3526 instances. The dataset is obviously too small for deep learning training. The high accuracy rate in the experimental results may be due to the uneven distribution and poor diversity of the test data.

Adebowale *et al.* [15] combined the convolutional neural network (CNN) and long short-term memory (LSTM) algorithm to classify phishing websites. The hybrid classifier obtained an accuracy of 93.28% and an average computational time of 25s by using image, frame and text features. They collected URLs from Phish Tank and Common Crawl and extracted image features from URLs. The image features are used to feed the offline CNN model, and the text features are contributed to the LSTM classifier. The innovative point of this solution is to combine the characteristics of pictures and text. However, from the experimental results, there is still room for improvement in the accuracy rate, and the

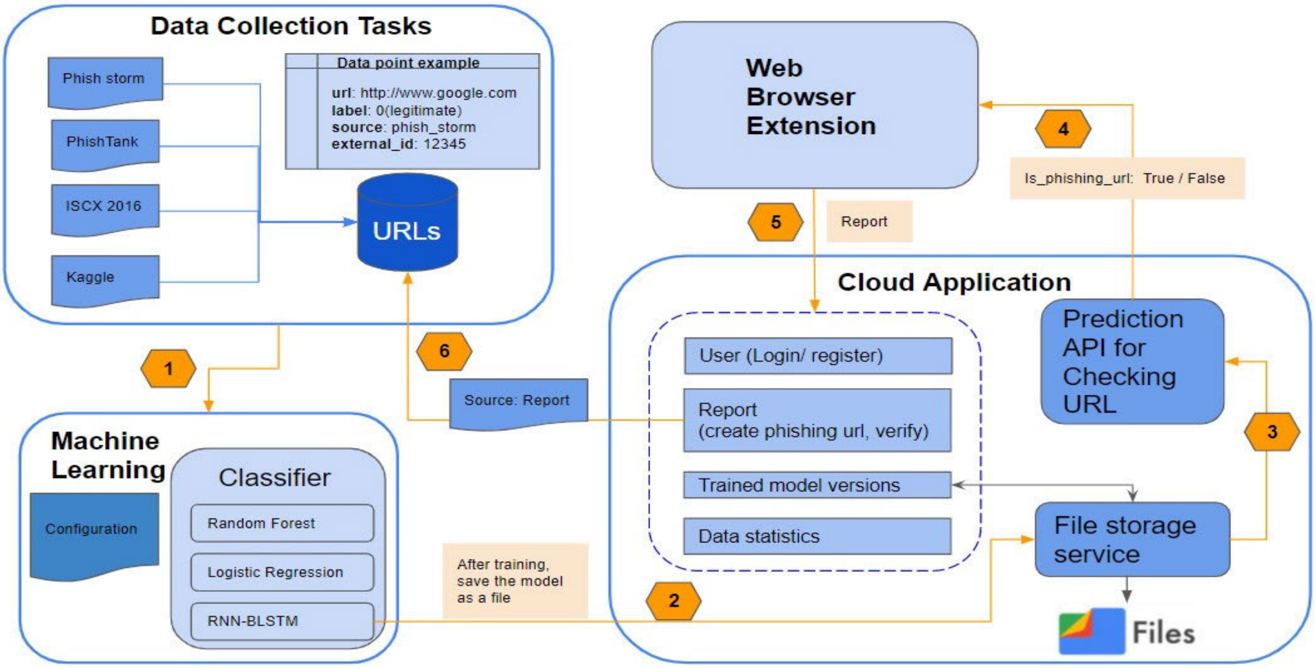


FIGURE 1. Architecture of the deep learning-based framework for detecting phishing URLs.

calculation time is too long to meet the requirements of real-time prediction products.

B. FRAMEWORKS AND SYSTEMS

When detecting whether a webpage is at risk of phishing attacks, the core service is a prediction service based on machine learning. The response time of predictive service is the most important indicator to measure the feasibility of this real-time system.

Atimorathanna *et al.* [16] introduced an anti-phishing protection system, which consists of a web browser extension, an e-mail detection plug-in, filters, and a machine learning-based phishing detecting server. The browser extension is used to extract the current URL, capture a screenshot, and store the user's visit history as a profile on the client-side. The server mainly uses the following processes to detect phishing links: (1) using the blacklist and whitelist of third-party services to filter new URLs; (2) using a machine learning model based on 13 features to predict whether the URL is a phishing link; (3) using computer vision technology to detect website logos and comparisons the similarity of screenshots of web pages. The logo detector in the article is used to identify 20 well-known online banks and some commonly used website logos.

The authors collected and established their own database for the training of the logo detection model and obtained an accuracy rate of more than 95%. The comparison of the similarity of the two screenshots uses Python's OpenCV library. The experimental results of the URL analyzer showed that the Random Forest classifier achieved the highest accuracy of 96.257%. It is a completed online real-time detection system for phishing, combining multiple methods to protect users from being attacked effectively. However, there is still room for improvement in the machine learning model's performance, and the number of logos that the logo classifier can detect is too small.

Maurya *et al.* [17] introduced an anti-phishing system, which contains a web browser extension. The browser plug-in obtains the current URL in real-time and extracts features based on the DOM structure, then detects whether there is a risk of phishing attacks and prompts the user. The detection service is divided into three stages, namely whitelist matching, blacklist filtering, and prediction based on a machine learning model. The prediction phase determines the URL that meets the criteria as a phishing link based on character-level features. For example, there are no hyperlinks to the web page, and the number of hyperlinks to external domain names exceeds a certain percentage. Such rules are vulnerable to attackers, and some normal URLs are likely to be misidentified. In addition, the author improves accuracy by combining three basic classification models.

Shah *et al.* [18] presented a machined learning-based browser extension for detecting phishing URLs. They trained the Random Forest model using the UCI dataset, which contains 11,055 instances with 30 normalized features. Therefore, it is required to extract features based on the current URL string in a real-time environment. In the article, the authors extracted 16 features that do not rely on third-party services. Experimental results show that the accuracy rate is 89.6%, which has a lot of room for improvement.

Sundaram *et al.* [19] built a Chrome browser extension for phishing websites detection. They used the UCI data set to train the model and packaged the trained model into a browser extension. The article did not describe the implementation details and results in detail, nor does it give the average calculation time for real-time detection of a URL. However, the feature extraction process relies on third-party domain name services.

Abiodun *et al.* [20] developed a website to verify a link is a phishing URL or not. The detector was implemented by JAVA programming language and a library named JSoup HTML Parser (JHP). This solution is mainly divided into three stages. The first is to use JSoup to parse the DOM structure of the website to be detected. The second is to analyze the number of link tag <a> from the DOM structure and analyze the attribute ''href'' value. The attribute value is classified as an empty link, external links and internal links. Third, the link calculator figured out an indicator, which has a value between 0 and 1. When the value exceeds 0.8, the URL to be verified is considered a phishing link. Since no machine learning model is introduced, there is no training process. In the experiment, the authors used 300 URLs to test the performance of the link calculator. The testing results showed they achieved 99.97% accuracy and a 0.03 false-negative rate. They will need to use a larger test data set to verify this solution in the future. From the analysis, it is a misjudgment to judge the phishing risk by analyzing the characteristics of the link tag from the website source code alone, and it is easy for attackers to use this rule to circumvent these rules.

A web browser architecture with an intelligent engine for phishing websites detection named EPDB is presented in [21]. Compared to the traditional web browser architectures, the EPDB has a brilliant engine-integrated machine learning model for detection in a real-time environment. They used the UCI dataset to train machine learning models. In the predictive process, the rule of extraction framework is applied, which could extract 30 features of a website. The experimental results showed the Random Forest classifier obtained the highest accuracy of 99.36%. Although the accuracy of the experimental data is very high, this solution also has some limitations and challenges. First, developing a browser is a highly complex task. Some functions of the browser need to be compatible with mature browser functions before they can be promoted to users. In addition, the data set for training the model is single, and the robustness of the model needs to be verified again. Finally, the rule-based feature extraction framework relies on third-party services.

III. FRAMEWORK DESIGN

Figure 1 depicts the architecture of the components of our proposed framework. There are four modules in terms of data collection tasks, machine learning (ML), cloud application, and web browser extension. The data collection module is an independent scheduled task application. The ML module is used for training modules. The web browser extension is a client-side product. The cloud application is built to deal with false alarms and phishing URLs reported by users from the web browser extension. The orange lines with arrows in the figure show the data interaction process.

The core process of this framework is mainly divided into the following six steps: the first is to collect and integrate data from various data sources; the second is to combine different data sets for machine learning model training, and store the trained model in a file system; the third is that the interface for predicting phishing risk calls the trained model to make predictions; the fourth is that the browser extension calls the prediction interface to perform real-time detection and display the detection results; the fifth is that users can submit real-time feedback when they disagree with the detection results, such as misjudgment, missed alarm; finally, the report submitted by the user is verified through manual review and automatic review strategy, and the verification result is synchronized to the data set

A. DATA COLLECTION TASKS

Data is the core of the field of machine learning. The quality and quantity of data significantly impact the performance of machine learning-based modules [22]. The data collection module is the foundation of this system. A data collecting task is divided into two parts, obtaining data from different data sources, then analyzing and storing data.

We collected data from different open sources shown in Table 1. The Phish Storm [2] dataset contains 96,018 URLs: 48,009 legitimate URLs and 48,009 phishing URLs. The ISCX-URL2016 [12] dataset contains 35378 legitimate URLs and 9965 phishing URLs. We loaded around 350,000 benign URLs from an open Kaggle project [23]. In addition, we initially collected 400,000 data and regularly grabbed new data from the Phish Tank platform [13] every day.

TABLE 1. Data sources.

Data source	Legitimate URLs	Phishing URLs
Phish storm [2]	48,009	47,902
Phish Tank [13]	0	178,495
ISCX-URL2016 [12]	35,378	9,965
Kaggle [23]	345,738	0

We analyzed the basic structure of the URL and parsed out basic information such as protocol, domain, subdomain, top-level domain, and path [24]. Table 2 presents the major fields of a table named URL. We stored data in a relational database, as it is flexible and efficient for providing data services by reading based on SQL. These data services can combine multiple data sets. For example, select 20,000 phishing links from phish tank and 20,000 good links from Kaggle, and combine them into a balanced data set with 40,000 instances.

B. MACHINE LEARNING

The machine learning module is mainly responsible for model training and model testing. In this framework, the data

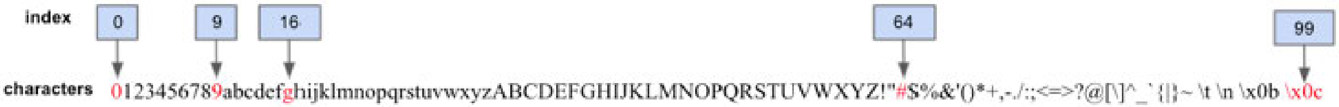

index
0
9
16
64
99
characters
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ \t \n \x0b \x0c

FIGURE 2. The characters dictionary with 100 ASCII characters which are widely used in URLs.

TABLE 2. The database table URL's structure.

name	description	sample
url	URL	https://amazom.mhmgmm.rest/mobile/
label	1: phishing 0: legitimate	1
source	Data source	Phish Tank
External id	Unique ID of the same data source	7270002
netloc	netloc	amazom.mhmgmm.rest
Gmt_created	Created date	2021-08-21 01:39:40

of the training model is updated regularly, and the training and testing processes of all models are automatically and regularly triggered. The system will record each run's parameters and data collection types and dump the model to the file storage system. It is flexible to add new models to the ML module. This research developed six machine learning models, namely Logistic Regression, Support vector machines (SVM), Random Forest, RNN, RNN-GRU, and RNN-Long short-term memory (LSTM).

1) PARAMETER CONFIGURATION

The parameter configuration process initializes the model parameters according to the configuration file. The configuration file includes a parameter grid corresponding to each model, and each parameter has a discrete number of values. In the model training process, one of the permutations and combinations of these parameter values will be selected for each training. When all the combinations are applied to the model and the training is completed, the optimal parameter combination can be obtained by comparing the accuracy of the model.

2) DATA LOADING

The dataset used for model training is obtained from the database through the data service. The data service supports the flexible selection of different data source combinations and datasets of varying data volumes. Each data instance contains a URL string and a label that signs the URL is a phishing link or legitimate link. The label values are normalized as 1 and 0.

3) FEATURE EXTRACTION

We treat the URL string as a document containing semantics and apply the Natural language processing (NLP) technology to extract features. The feature extraction process converts a collection of text documents to a matrix of token counts, and each token stands for one word. In classical machine learning models, the tokenization process converts a URL string to a list of words. Therefore, the number of features equals the vocabulary size found by analyzing the data.

In deep learning models, the tokenization process parses a URL string to a list of characters (Character-level tokens). The characters in the URL come from the ASCII character set. We chose the most common 100 characters as the character set dictionary for this study. Figure 2 shows all the arranged characters and the corresponding index.

The maximum length of a URL is 2083 characters [24]. Because of the calculation time of the deep learning model and the analysis of the statistical data of the existing data set, we set the maximum number of URL characters to 200. Therefore, each URL can be transformed into a 200*100 matrix. The position of the dictionary corresponding to each character is marked as 1, and the remaining values are 0. Figure 3 shows the process of forming a matrix using Google's official website as an example.

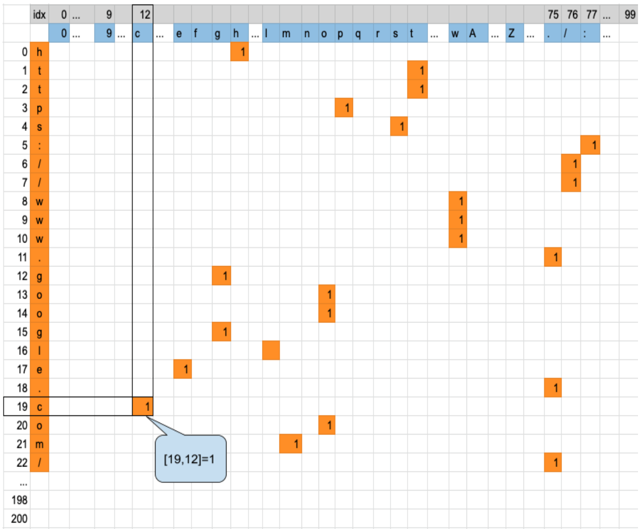


FIGURE 3. The process of creating a feature matrix from a URL string and the character dictionary.

4) MODELLING

It is a solution to treat a URL as a document and use character separators to parse words as features. However, many words in URLs also lack semantics. Moreover, the analysis of word-level results in an extensive dictionary will slow the calculation time. Therefore, we choose to analyze with character level and the characters of the entire URL as a sequence. The recurrent neural network (RNN) is a feedback neural network that stores temporary states. It's suitable for training sequence data [25]. Figure 4 shows a regular RNN architecture that consists of an input layer, several hidden layers and an output layer. Compared to the feedforward artificial neural networks (ANN), RNNs have a unique architecture with a connection function between neurons in hidden layers. The figure shows that the current hidden state is related to the previous hidden state and the current input. The current hidden state's functional form can be represented as Eq. (1) and (2). The tanh is a nonlinear function, W represents the weights between the neurons, and b is the bias vector of the setting. The softmax calculates the output value as an activation function, as shown in Eq. (3), and the model prediction value is related to the current hidden state.

$$h_t = f_W\left(h_{t-1}, x_t\right) \tag{1}$$

$$h_t = tanh\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right) \tag{2}$$

$$Y_t = softmax\left(W_{yh}h_t + b_y\right) \tag{3}$$

The scenario that detects the phishing link is a many-to-one task type, the input is character-level sequence data, and the output is a category. Figure 5 shows the structure of one hidden layer.

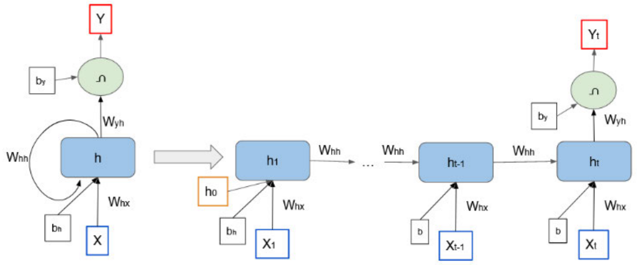


FIGURE 4. **The architecture of a basic RNN.** W_{hx}, W_{hh}, W_{yh} **respectively means the weight matrix between input and hidden layer, the weight matrix between two hidden layers, the weight matrix between hidden and output layer.**

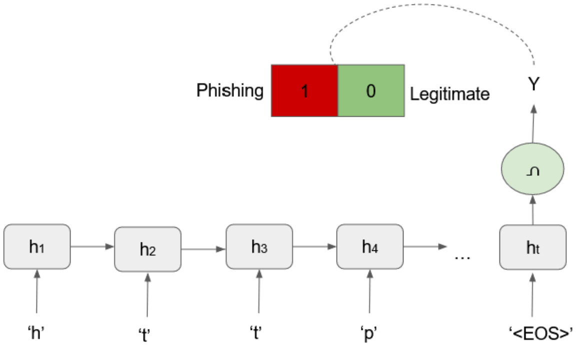


FIGURE 5. **Character-level features in an RNN model for phishing URL classification.**

Before model training, the structure of the model is fixed, and activation function is established, so the process of model training is the process of optimizing the weights parameters by calculating each error. First, randomly initialize the weights matrix, then calculate the difference between the actual value and the predicted value, then use the optimized algorithm to find the optimal solution to minimize the difference, and finally adjust each weight by calculating the step each time.

Depending on the architecture of RNN and activation functions used, the basic RNN architecture does not perform well for handling inputs for long sequences because of the vulnerability to gradient vanishing or exploding problems [26]. To address these, Hochreiter and Schmidhuber introduced a gradient-based model named long short-term memory (LSTM) in 1997 [27]. They invented a long short-term memory unit instead of tanh function to compute hidden states. The LSTM unit consists of three gates and two memory cells. Cho *et al.* proposed a novel model with a hidden unit, which was motivated by LSTM in 2014 [28].

Since the hidden unit contains two gates to control and calculate the hidden state, this model is also named gated recurrent unit (GRU). Figure 6 demonstrates the architecture of the gate units. It can be said that long short-term memory network (LSTM) and gated recurrent unit (GRU) are two enhanced versions of RNN. Many studies and experimental data show that for sequence data training, the LSTM and GRU architecture can achieve better performance than the basic RNN architecture [29]–[31].

5) OPTIMIZER AND LOSS FUNCTION

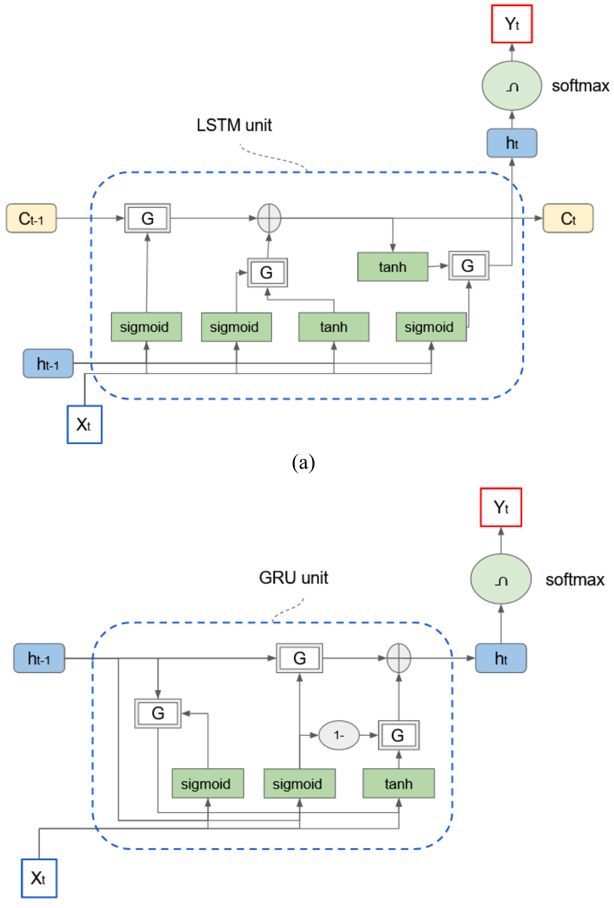
In the model training process, it is also essential to choose a suitable optimizer and loss function. Among many optimization algorithms, we have selected Adam, which is a popular and effective optimization algorithm for deep learning [32]. Since the problem is a binary classification problem, we used the cross-entropy loss function [33], which is also called log loss function. According to the scenario of the current problem, the output predicted value is a floating-point number between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. It is believed that it will converge quickly in the initial stage of training with the same learning rate. The loss value of each epoch is calculated by calculating the average loss value of all data points.

6) DUMP MODEL TO FILE SYSTEM

After the model training is completed, the system will save the model to the specified file directory. Each file has a different version number, and the file name is related to the version. When the system is deployed to the cloud, it will use the Google file system to manage these files. The prediction service can load these models directly and perform real-time prediction for single or multiple URL links, and each call takes less than 200ms.

C. WEB BROWSER EXTENSION

We have developed a Chrome browser extension. Since users installed the extension in the Chrome browser, the extension



(a)

(b)

FIGURE 6. (a). A LSTM unit. G stands for a gate. h_t means the current hidden state, and C_t means a current memory cell state. The t-1 is a previous time. (b). A GRU unit. G stands for a gate. h_t means the current hidden state, and h_{t-1} means the previous hidden state.

will automatically detect whether the newly opened URL is at risk of phishing. If there is a risk, the user will be interactively prompted through a popup box, and the entrance feedback error detection will be provided. The extension will call the HTTP interface of the prediction service to obtain the detection result and save the detection result in Chrome's storage.

D. CLOUD APPLICATION

When a false alarm or missed alarm occurs in the prediction service, the user can take the initiative to report the current falsely detected URL from the browser plug-in portal. We have developed a website to receive these reports. Once the report is submitted to the system, the system has a manual review process to confirm the risks of these URLs. In addition, there are automatic audit strategies to improve audit efficiency. Once the review is completed, these URLs will be regularly synchronized to the data collection module, and the source is reported. In addition, the website provides a detection interface for browser plug-ins, supports multi-strategy detection, and currently includes whitelists, blacklists, and machine learning models.

IV. PROTOTYPE IMPLEMENTATION

The prototype implementation of the entire framework is divided into three independent applications.

The browser extension is independently packaged and uploaded to the Chrome browser according to the extension development specifications of the Chrome browser and will be reviewed and released by the Chrome platform. Chrome browser plug-in development uses three web front-end development languages: HTML, JavaScript, and CSS.

The data collection application is based on Python as the main development language, using scheduled tasks to manage the collection tasks of each data source. Among them, phish tank crawls information from web pages, using the most used package named Beautiful Soup.

Model training, prediction services and product official website are integrated into one application. This application also uses Python as the primary language and imports Flask as the web framework. Model training is managed by timed tasks as well. After the training is completed, the core performance indicators are written into the MySQL database in real-time, and the model is dumped into the file system. The prediction service is a RESTful API that provides clients with real-time POST requests to obtain detection results. The core function of the official website is to accept the suspected phishing link submitted by the user and determine the link risk by manual and automatic verification.

A. WEB FRAMEWORK

We used Python as our core language, which is a modern high-level programming language in the field of data mining and machine learning. There are various frameworks and libraries for the Python language. In our system, data collection, data storage, model training, websites, and HTTP services are all supported by mature libraries and frameworks. In addition, the access and use of these packages are very simple and convenient.

Considering the usage scenarios and read and write performance, we chose the MySQL relational database. First, the website has user management, report management, model version management and other functions, which require a relational database. In addition, the data set used for model training is acquired dynamically. It is very flexible to combine different data sources and data volumes to form a new data set for model training. For example, we obtained 200,000 phishing URLs from Phish tank and 340,000 legitimate URLs from Kaggle. A balanced data set with 40,000 URLs can be flexibly combined, including 20,000 phishing URLs and 20,000 legal URLs.

We imported the Flask as a web framework to provide HTTP service and maintain the official website. Flask is a lightweight web framework and easy to extend [34]. For example, the flask-user package provides user authorization services.

B. TRAINING MODEL

1) SCIKIT-LEARN

The scikit-learn is open-source and widely used for predictive data analysis in the machine learning field [35]. We imported a scikit-learn library to train three traditional machine learning models: Logistic Regression, Random Forest, and support vector machine.

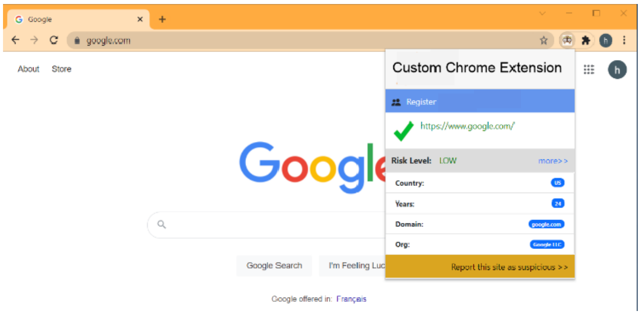
2) PYTORCH

The PyTorch is an open-source deep learning framework and development platform. We used the dataset module to build a custom dataset as input for the training model. In the deep learning models' construction, we imported the linear layer, RNN layer, GRU layer, and LSTM layer. We imported torch.cuda package that utilizes GPUs for parallel computation [36].

C. WEB BROWSER EXTENSION

The development of Chrome browser extension must strictly follow the development guidelines in [37]. The chrome plug-in has a configuration to contract version numbers, introduce built-in APIs, and permission control. Data interactions between modules are messaged and temporarily stored in Chrome storage. A background script listens for browser tab change events, gets the URL currently accessed, calls the back-end prediction service to get the result, and finally sends the result to a content script. The content script is primarily responsible for presenting the results on the page. In addition, a popup Html shows the details.

Figure 7 shows an example of entering a legitimate URL. In this case, the user opens the page in a web browser with no additional information. When the user clicks the plug-in button on the right side of the toolbar, the popup page is displayed with the current URL string, risk level, and other information.



https://www.google.com/

Risk Level: LOW

Country: US

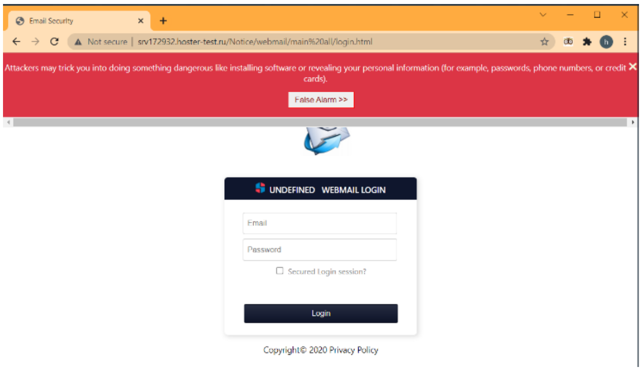
Years: 24

Domain: google.com

Org: Google LLC

FIGURE 7. A screenshot of the chrome browser extension's popup page. The current URL is 'https://www.google.com/'.

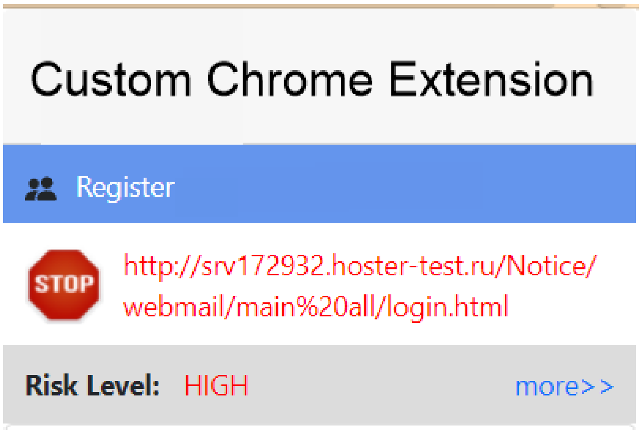
Figure 8 presents an example. When the entered URL is detected as a phishing link, a popup box with a red background appears on the page, prompting the user that the website is at phishing risk. If the user confirms that the URL is not a phishing network, they can click the false alarm button to respond to this false alarm. Figure 9 shows the style and content of popup pages on high-risk sites.


Email Security
Not secure | srv172932.hoster-test.ru/Notice/webmail/main%20all/login.html
False Alarm >>
UNDEFINED WEBMAIL LOGIN
Email
Password
Login
Copyright© 2020 Privacy Policy

Attackers may trick you into doing something dangerous like installing software or revealing your personal information (for example, passwords, phone numbers, or credit cards).

□ Secured Login session?

FIGURE 8. A screenshot of the chrome browser extension's alert warning message window when it detected a phishing URL. The current URL is "http://srv172932.hoster-test.ru/Notice/webmail/main%20all/login.html".



http://srv172932.hoster-test.ru/Notice/webmail/main%20all/login.html

Risk Level: HIGH

FIGURE 9. A screenshot of the chrome browser extension's popup page when it detected a phishing URL.

V. EVALUATION RESULTS

All the experiments were executed on a MacBook Pro 2020 running Quad-Core Intel Core i5 CPU @ 2 GHz with macOS Big Sur 11.5.2 operating system. The server has a 500 GB storage capacity. In addition, the test data ratio is 0.2.

We have seven data sets, six models, and each model has different parameters. Our experiment is carried out in the following steps to find the optimal model quickly. First, choose a model that may perform well from the theoretical analysis. We prefer the GRU model. Then, we compare the data sets with the best performance from the experimental results. Second, we use the data set determined in the previous experiment to train different models and compare the results.

Finally, we optimize the model hyperparameters for the model with the dataset. The primary method is to enumerate the optional discrete values of the parameters and perform cross-combination to compare the performance of all experimental results.

Evaluate whether a machine learning model has high performance, standard statistical metrics with accuracy, recall, and precision [38]. These indicators are obtained by simple mathematical calculations of four atomic statistical indicators in terms of the number of correctly identified positive instances (TP), the number of correctly identified negative data points (TN), the number of negative data points predicted by the model are positive (FP), the number of positive instances labelled as negative (FN). In the article, we use the F1 score to represent the meaning of the recall and precision. In addition, in cybersecurity detection applications, false alarms can affect the user experience and trust, and leak alarms are likely to directly cause user losses. Therefore, we use accuracy, F1, false-positive rate, false-negative rate to measure the efficiency of models. The mathematical formulas for these metrics are as Equations (4), (7), (8), and (9).

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$\begin{aligned} \text{F1} &= \frac{2 \times precision \times recall}{precision + recall} \\ &= \frac{TP}{TP + \frac{1}{2}(FP + FN)} \end{aligned} \tag{7}$$

$$\text{false positive rate} = \frac{FP}{FP + TN} \tag{8}$$

$$\text{false negative rate} = \frac{FN}{FN + TP} \tag{9}$$

Furthermore, Average precision (AP) is a widely used metric in evaluating the accuracy of deep learning models by computing the average precision value for recall value over 0 to 1; higher is better. Mean average precision (mAP) is the average of AP. Equation (10) shows the calculation logic. In this scene, the number of classes is two.

$$\text{mAP} = \frac{1}{classes} \sum_{c \in classes} \frac{TP(c)}{TP\,(c) + FP(c)} \tag{10}$$

A. GRU WITH DIFFERENT DATASETS

In this experiment, we compared different datasets fed to the GRU classifier. The number of epochs is 20, the batch size is 32, KPT stands for the data collected from Kaggle and Phish Tank, and each KPT dataset is a balanced dataset, which consists of the same number of phishing URLs and legitimate URLs. Table 3 shows the core performance indicators of the GRU model with 8 datasets. The ISCX dataset obtained the highest accuracy. However, the F1 score is lower than the other three KPT datasets, and the false-negative rate is high. In other words, more legitimate instances are predicted as phishing URLs during the test data process. This result is probably because there are not enough data points labelled as phishing.

Furthermore, we combined false-positive rate and false-negative rate to measure efficiency. From figure 10, we can see that the RNN-GRU model with the KPT-12 dataset performs best. In KPT datasets, the false rate decreases linearly as the number of data increases.

TABLE 3. The GRU model with different datasets.

Dataset	accuracy	F1	False-positive rate	False-negative rate	mAP
Phish storm: 95,911 URLs (Legitimate URLs: 48,009; Phishing URLs: 47,902)	0.9758	0.9748	0.0269	0.0212	0.960
ISCX: 45,343 URLs (Legitimate URLs: 35378; Phishing URLs: 9965)	0.9947	0.9874	0.0011	0.0191	0.986
KPT-4: 40,000 URLs	0.981	0.980	0.0091	0.0285	0.948
KPT-6: 60,000 URLs	0.9882	0.988	0.0089	0.0145	0.967
KPT-8: 80,000 URLs	0.9898	0.9894	0.0134	0.0073	0.973
KPT-10: 100,000 URLs	0.9906	0.9904	0.006	0.0132	0.984
KPT-12: 120,000 URLs	0.9918	0.9915	0.0104	0.0059	0.986

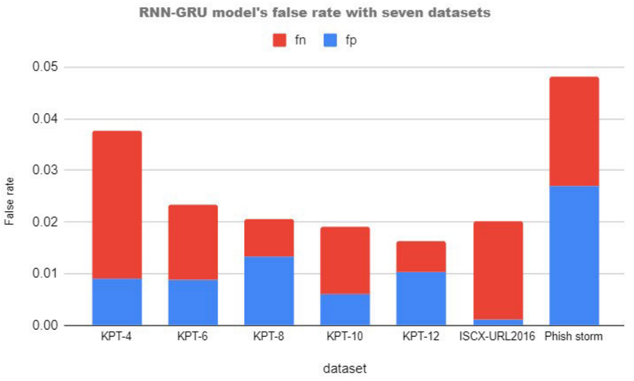


FIGURE 10. The RNN-GRU model's performance in experiments with different datasets.

Figure 11 shows the accuracy and F1 of each dataset. The KPT-12 dataset obtained 99.18% accuracy and 99.15% F1 score. To quantify how well the RNN-GRU model performs every class, we calculate the mean of average precision (mAP) of a set of classes. The mean average precision (mAP) in RNN-GRU models with KPT-12 dataset is 0.986.

Assessing the efficiency of a machine learning model is incomplete, depending on accuracy. In experiments, it is customary to get high accuracy in one dataset and not perform well in another. It is also likely that models with high accuracy will not predict new data accurately in a real-time

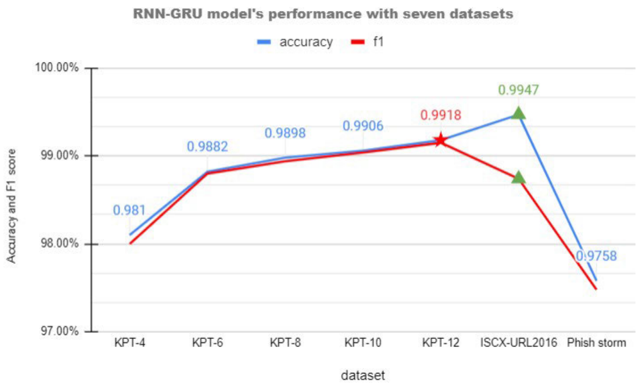


FIGURE 11. The accuracy and F1 score of different datasets applied to an RNN-GRU model.

environment. These situations may be due to the fact that overfitting has already occurred [39]. Overfitting is a concept in data mining that analyzes whether a trained model can efficiently predict unknown new data [40]. In machine learning-based classification models, it is common to compare errors in the training process with errors in the validation process to see if there is overfitting, along with epoch. Figure 9 presents the training loss and validation loss along with epochs in the RNN-GRU model. One of the strategies to avoid overfitting is early-stopping [40], [41]. As shown in figure 12, the epoch equals 6 is the demarcation point between underfitting and overfitting.

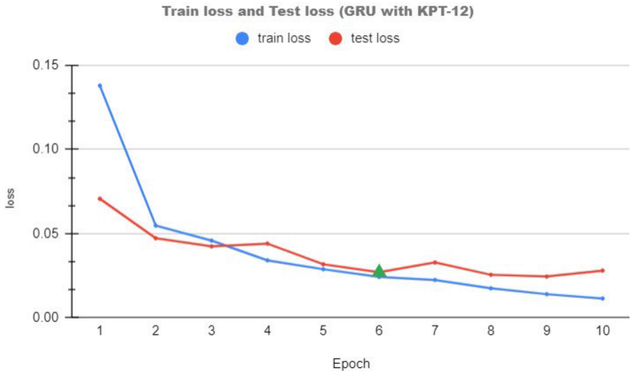


FIGURE 12. The GRU's training loss and testing loss along with epochs.

B. THE KPT-12 DATASET FOR DIFFERENT CLASSIFIERS

The experiment is to apply the same data set to different machine learning models, from which the best performance model can be analyzed. Many studies have shown that the random forest classifier performs better than other traditional classification models in detecting phishing networks [7], [8], [42]. Therefore, we chose logistic regression, SVM, and random forest. In addition, the RNN model architecture is ideal for training sequence data in deep learning algorithms. In this experiment, we compared the performance metrics of these six models. Table 4 presents that the RNN-GRU achieved the highest accuracy of 99.18%, and the Random Forest obtained the lowest false-positive rate of 0.0047%.

TABLE 4. Different classifiers' performance comparison by the same dataset: KPT-12.

Classifier	accuracy	F1	False-positive rate	False-negative rate
Logistic Regression	0.9889	0.9888	0.0081	0.0141
Support vector machine (SVM)	0.9885	0.9885	0.01	0.0129
Random Forest	0.985	0.9849	0.0047	0.0253
RNN	0.7412	0.7089	0.1813	0.3372
RNN-GRU	0.9918	0.9915	0.0104	0.0059
RNN-LSTM	0.9895	0.9891	0.0118	0.0089

In this experiment, the accuracy and F1 scores of all models were very close. This performance can be seen in Figure 13. Because the accuracy of the underlying RNN model is less than 0.9, it is not shown in the figure. From the results data of the three deep learning models, the effects of gate unit and LSTM unit on sequence data training are explained once again.

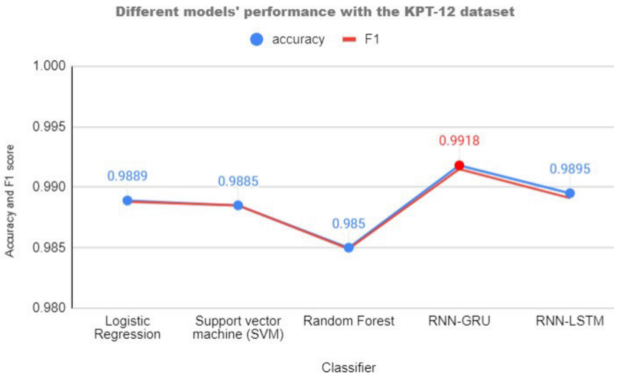


FIGURE 13. The accuracy and F1 score in different models with KPT-12 dataset.

Although the random forest model gets the lowest false-positive rate, the false-negative rate is the highest, and the sum of the two error rates is the highest.

C. HYPERPARAMETER OPTIMIZATION

From the above two experimental results, we can conclude that the KPT-12 dataset applied to the RNN-GRU model can obtain the best performance.

The third experiment is to optimize the model hyperparameters for better performance. The optional values of parameters are listed in table 5. A total of 162 combinations of optional values for all parameters will be performed in turn. Because the computer GPU running the experiment does not support parallel computing and it takes a long time to train

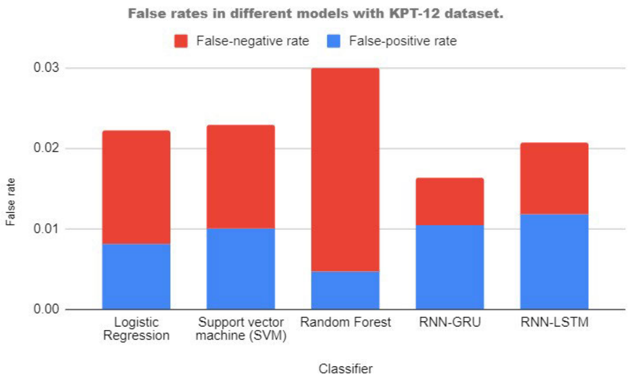


FIGURE 14. The false-positive rate and false-negative rate in different models with KPT-12 dataset.

a model with the KPT-12 dataset, the experiment will be performed after the system is deployed to the cloud. Access the TensorBoard tool to visualize a comparison of execution results and performance metrics to get the best combination of parameters [43].

TABLE 5. The optional values of parameters.

Parameter	Optional value	Default value
Batch size	[32, 64, 128]	32
epoch	[6, 10, 20]	10
shuffle	[true, false]	true
Learning-rate	[0.01, 0.005, 0.001]	0.001
Layer number	[1,2,3]	2

D. COMPARISON

This section compares the RNN-GRU model to existing solutions that train deep learning models to detect phishing websites. Table 6 shows a comparison from different dimensions, such as data collection, models, performance indicators, limitations.

As for the limitations of the proposed solution implementation, since there are no short links in the data set of the training model, all current prediction services cannot accurately detect whether short links are at risk of phishing. Furthermore, we intercepted the first 200 characters of the URL, so for URLs with more than 200 characters, part of the information is lost, so it may affect the detection results. In addition, the process of the automatic review report is currently judged based on rules such as remote IP address, client information, and the number of times the URL has been submitted. This strategy can easily be used maliciously by phishing attackers. In the future, more data will be needed to support automatic review results, for example, by obtaining the HTML of the current URL, identifying the similarity between the logo image and the whitelisted website, and whether there is an input box in the HTML.

TABLE 6. Comparison of proposed RNN-GRU model with other deep learning-based solutions.

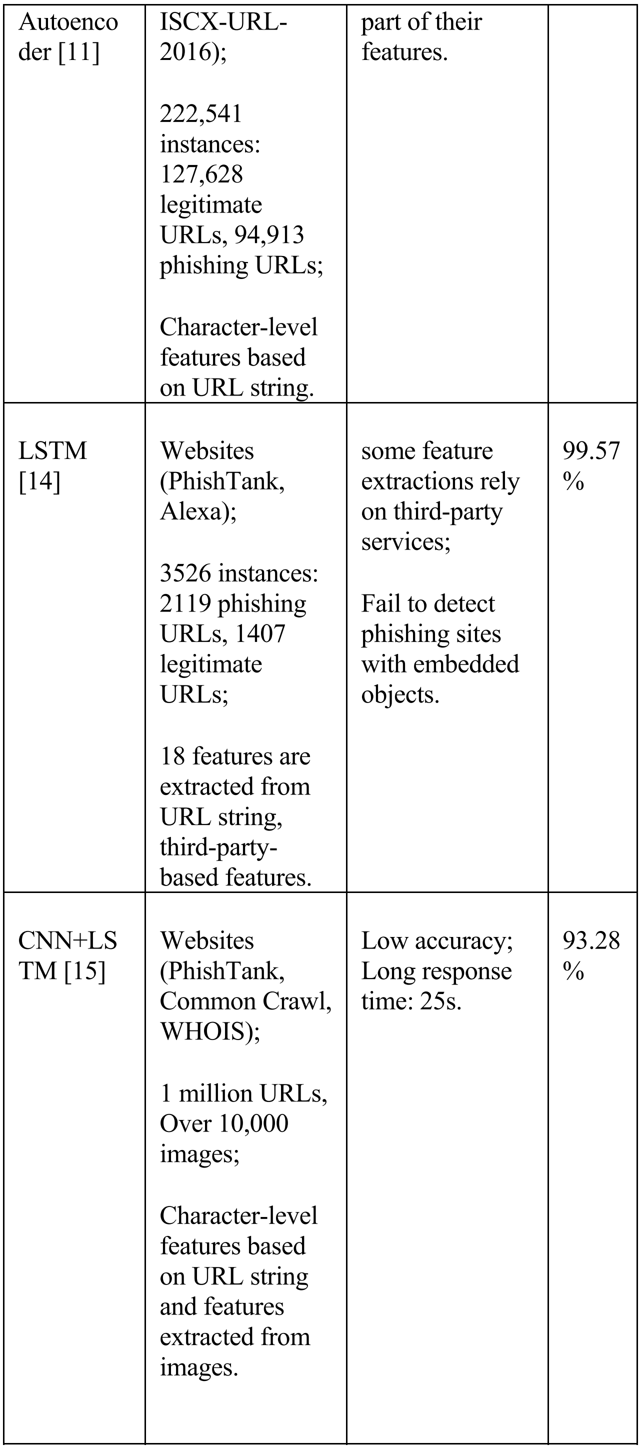
Model or Algorithm	Dataset	Limitations	Accuracy

RNN-GRU	Websites (PhishTank, Kaggle);
	120,000 instances:60,000 phishing URLs, 60,000 Legitimate URLs;
	Character-level features based on URL string.
		Short URLs are not supported;
		URLs of more than 200 characters will lose some of their features
			99.18 %

Transfer Learning [44]	Website (Huawei Symantec);
	177,417 instances: 36,560 phishing URLs, 14,0857 legitimate URLs;
	15 features based on URL string, domain, and sensitive words
		Some feature extractions rely on third-party services.	97%

Reinforcement Learning [45]	Website (PhishTank, Yandex Search engine); 73,575 instances:37,175 phishing URLs, 36,400 legitimate URLs. 14 features based on URL string, domain, and HTML	Low accuracy; some feature extractions rely on third-party services.	90.1 %
Convolutional	Websites (PhishTank, PhishStorm,	Some long URLs will lose	97.82 %

TABLE 7. ***(Continued.)*** **Comparison of proposed RNN-GRU model with other deep learning-based solutions.**



Autoencoder [11]	ISCX-URL-2016);
	222,541 instances: 127,628 legitimate URLs, 94,913 phishing URLs;
	Character-level features based on URL string.
		part of their features.	

LSTM [14]	Websites (PhishTank, Alexa);
	3526 instances: 2119 phishing URLs, 1407 legitimate URLs;
	18 features are extracted from URL string, third-party-based features.
		some feature extractions rely on third-party services;
		Fail to detect phishing sites with embedded objects.
			99.57 %

CNN+LSTM [15]	Websites (PhishTank, Common Crawl, WHOIS);
	1 million URLs, Over 10,000 images;
	Character-level features based on URL string and features extracted from images.
		Low accuracy; Long response time: 25s.	93.28 %

VI. CONCLUSION AND FUTURE WORK

Many machine learning-based solutions have been proposed in recent years to deal with phishing attacks, but results have not been verified in live browsing environments, and there is a lack of analysis and research of products for phishing detection. In this paper, we proposed a framework for phishing detection in a real-time browsing environment. The novel features of the framework are:

1) We utilized closed-loop data to drive better performance of machine learning models. A dataset is fundamental to model training, and high-quality data can improve the performance of a model. The feedback data from users are high-quality data with advancement, accuracy, and sensitivity.
2) The system is running in a real-time environment without delays. The prediction results are displayed when the web page is opened.
3) Experimental data can be tracked. The model training process is an automated task, and each execution result is stored in a real-time database.
4) We have developed a browser extension as a client product that every ordinary netizen can use.
5) The implementation of predictive services is extendable, and individual detection services can be combined. For example, you can introduce a blacklist filtering service, computer vision service.
6) The feature extraction process in the deep learning model is independent of third-party services.

In the future, we will deploy the whole system to a cloud platform. Configure machines with NVIDIA GPUs for model training and increase efficiency with GPU's parallel computing power. Afterward, users can download the extension through the Chrome Web Store. In addition, we plan to implement our framework as a plug-in for other browsers.

REFERENCES

[1] L. Tang and Q. H. Mahmoud, "A survey of machine learning-based solutions for phishing website detection," *Mach. Learn. Knowl. Extraction*, vol. 3, no. 3, pp. 672–694, Aug. 2021, doi: 10.3390/make3030034.

[2] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 4, pp. 458–471, Dec. 2014.

[3] (Jun. 2021). *Phishing Activity Trends Report 1st Quarter 2021*. APWG. Accessed: Oct. 20, 2021. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2021.pdf

[4] (2020). *2020 Internet Crime Report*. [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf

[5] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Comput. Appl.*, vol. 25, no. 2, pp. 443–458, Nov. 2013, doi: 10.1007/s00521-013-1490-z.

[6] M. A. El-Rashidy, "A smart model for web phishing detection based on new proposed feature selection technique," *Menoufia J. Electron. Eng. Res.*, vol. 30, no. 1, pp. 97–104, Jan. 2021, doi: 10.21608/mjeer.2021.146286.

[7] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, and X. Chang, "A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment," *Comput. Commun.*, vol. 175, pp. 47–57, Jul. 2021, doi: 10.1016/j.comcom.2021.04.023.

[8] E. Gandotra and D. Gupta, "Improving spoofed website detection using machine learning," *Cybern. Syst.*, vol. 52, no. 2, pp. 169–190, Oct. 2020, doi: 10.1080/01969722.2020.1826659.

[9] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," *Secur. Commun. Netw.*, vol. 2019, pp. 1–15, Oct. 2019, doi: 10.1155/2019/2595794.

[10] M. Sabahno and F. Safara, "ISHO: Improved spotted hyena optimization algorithm for phishing website detection," *Multimedia Tools Appl.*, Mar. 2021, doi: 10.1007/s11042-021-10678-6.

[11] S.-J. Bu and S.-B. Cho, "Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing URL detection," *Electronics*, vol. 10, no. 12, p. 1492, Jun. 2021, doi: 10.3390/electronics10121492.

[12] *URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. Accessed: Oct. 20, 2021. www.unb.ca. [Online]. Available: https://www.unb.ca/cic/datasets/url-2016.html

[13] *PhishTank > See All Suspected Phish Submissions*. Accessed: Oct. 20, 2021. www.phishtank.com.[Online]. Available: https://www.phishtank.com/phish_archive.php

[14] M. Somesha, A. R. Pais, R. S. Rao, and V. S. Rathour, "Efficient deep learning techniques for the detection of phishing websites," *Sādhanā*, vol. 45, no. 1, pp. Jun. 2020, doi: 10.1007/s12046-020-01392-4.

[15] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, "Intelligent phishing detection scheme using deep learning algorithms," *J. Enterprise Inf. Manage.*, to be published. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/JEIM-01-2020-0036/full/html, doi: 10.1108/jeim-01-2020-0036.

[16] D. N. Atimorathanna, T. S. Ranaweera, R. A. H. Devdunie Pabasara, J. R. Perera, and K. Y. Abeywardena, "NoFish; total anti-phishing protection system," in *Proc. 2nd Int. Conf. Advancements Comput. (ICAC)*, Dec. 2020, pp. 470–475, doi: 10.1109/ICAC51239.2020.9357145.

[17] S. Maurya, H. Singh, and A. Jain, "Browser extension based hybrid anti-phishing framework using feature selection," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 11, pp. 1–10, 2019, doi: 10.14569/ijacsa.2019.0101178.

[18] B. Shah, K. Dharamshi, M. B. Patel, and V. Gaikwad. (2020). *Chrome Extension for Detecting Phishing Websites*. Semantic Scholar. [Online]. Available: https://www.semanticscholar.org/paper/Chrome-Extension-for-Detecting-Phishing-Websites-Shah-Dharamshi/fa99621bdc27cbd32ed799d7a6c1848ac644e8a8

[19] K. M. Sundaram, R. Sasikumar, A. S. Meghana, A. Anuja, and C. Praneetha, "Detecting phishing websites using an efficient feature-based machine learning framework," *Revista Gestão Inovação e Tecnologias*, vol. 11, no. 2, pp. 2106–2112, Jun. 2021, doi: 10.47059/revistageintec.v11i2.1832.

[20] O. Abiodun, A. S. Sodiya, and S. O. Kareem, "Linkcalculator—An efficient link-based phishing detection tool," *Acta Inf. Malaysia*, vol. 4, no. 2, pp. 37–44, Oct. 2020, doi: 10.26480/aim.02.2020.37.44.

[21] M. G. Hr, M. V. Adithya, and S. Vinay, "Development of anti-phishing browser based on random forest and rule of extraction framework," *Cybersecurity*, vol. 3, no. 1, pp. 1–14, Oct. 2020, doi: 10.1186/s42400-020-00059-1.

[22] N. Gupta, S. Mujumdar, H. Patel, S. Masuda, N. Panwar, S. Bandyopadhyay, S. Mehta, S. Guttula, S. Afzal, R. Sharma Mittal, and V. Munigala, "Data quality for machine learning tasks," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 4040–4041, doi: 10.1145/3447548.3470817.

[23] S. Kumar. (2019). *Malicious and Benign URLs*. kaggle.com. Accessed: Oct. 20, 2021. [Online]. Available: https://www.kaggle.com/siddharthkumar25/malicious-and-benign-urls

[24] (2020). *URL Structure [2020 SEO Best Practices]*. Moz. [Online]. Available: https://moz.com/learn/seo/url

[25] S. Smys, J. I. Zong Chen, and S. Shakya, "Survey on neural network architectures with deep learning," *J. Soft Comput. Paradigm*, vol. 2, no. 3, pp. 186–194, Jul. 2020, doi: 10.36548/jscp.2020.3.007.

[26] S. Seo, C. Kim, H. Kim, K. Mo, and P. Kang, "Comparative study of deep learning-based sentiment classification," *IEEE Access*, vol. 8, pp. 6861–6875, 2020, doi: 10.1109/ACCESS.2019.2963426.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[28] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[29] J. Chung, C. Gulcehre, and K. Cho. (Dec. 2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Accessed: Aug. 29, 2021. [Online]. Available: https://ashutoshtripathicom.files.wordpress.com/2021/06/paper-on-rnn-and-lstm-sequential-modelling.pdf

[30] A. Khan and A. Sarfaraz, "RNN-LSTM-GRU based language transformation," *Soft Comput.*, vol. 23, no. 24, pp. 13007–13024, Aug. 2019, doi: 10.1007/s00500-019-04281-z.

[31] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 235–245, Oct. 2019, doi: 10.2478/jaiscr-2019-0006.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[33] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020, doi: 10.1109/ACCESS.2019.2962617.

[34] *Welcome to Flask—Flask Documentation (2.0.x)*. flask.palletsprojects.com. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/

[35] (2019). *Scikit-Learn: Machine Learning in Python—Scikit-Learn 0.20.3 Documentation*. Scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/index.html

[36] *Torch.Cuda—Pytorch 1.9.1 Documentation*. Pytorch.org. Accessed: Oct. 14, 2021. [Online]. Available: https://pytorch.org/docs/stable/cuda.html

[37] Welcome. (Nov. 9, 2020). *Chrome Developers*. Accessed: Oct. 14, 2021. [Online]. Available: https://developer.chrome.com/docs/extensions/mv3/

[38] V. Babel, K. Singh, S. K. Jangir, B. Singh, and S. Kumar. (2019). *Journal of Analysis and Computation (JAC) Evaluation Methods for Machine Learning*. Accessed: Oct. 14, 2021. [Online]. Available: http://www.ijaconline.com/wp-content/uploads/2019/06/ICITDA_2019_paper_88-.pdf

[39] H. N. A. Pham and E. Triantaphyllou, "The impact of overfitting and overgeneralization on the classification accuracy in data mining," in *Soft Computing for Knowledge Discovery and Data Mining*, O. Maimon and L. Rokach, Eds. 2008, pp. 391–431, doi: 10.1007/978-0-387-69935-6_16.

[40] IBM Cloud Education. (Mar. 3, 2021). *What is Overfitting*. www.ibm.com. [Online]. Available: https://www.ibm.com/cloud/learn/overfitting

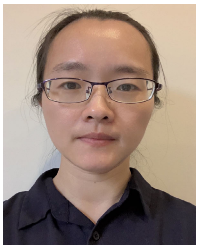
[41] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, Feb. 2019, Art. no. 022022, doi: 10.1088/1742-6596/1168/2/022022.

[42] G. H. Lokesh and G. BoreGowda, "Phishing website detection based on effective machine learning approach," *J. Cyber Secur. Technol.*, vol. 5, no. 1, pp. 1–14, Jan. 2021, doi: 10.1080/23742917.2020.1813396.

[43] *Visualizing Models, Data, and Training With Tensorboard—Pytorch Tutorials 1.9.1+Cu102 Documentation*. Pytorch.org. Accessed: Oct. 15, 2021. [Online]. Available: https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html

[44] J. Zhang, Y. Ou, D. Li, and Y. Xin, "A prior-based transfer learning method for the phishing detection," *J. Netw.*, vol. 7, no. 8, p. 1201, Aug. 2012, doi: 10.4304/jnw.7.8.1201-1207.

[45] M. Chatterjee and A.-S. Namin, "Detecting phishing websites through deep reinforcement learning," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2019, pp. 227–232.



LIZHEN TANG received the bachelor's degree in mechanical engineering and automation from Zhejiang Sci-Tech University, Zhejiang, China. She is currently pursuing the M.A.Sc. degree in electrical and computer engineering with Ontario Tech University. She worked as a Software Developer at Alibaba Group, Hangzhou, China, for eight years. Her research interests include machine learning, neural networks, cybersecurity, and data science.

QUSAY H. MAHMOUD (Senior Member, IEEE) was the Founding Chair at the Department of Electrical, Computer and Software Engineering, Ontario Tech University, Canada. He has worked as an Associate Dean with the Faculty of Engineering and Applied Science, Ontario Tech University. He is currently a Professor of software engineering. His research interests include intelligent software systems and cybersecurity.