Real-time phishing detection using deep learning methods by extensions

Article *in* International Journal of Electrical and Computer Engineering (IJECE) · June 2024

DOI: 10.11591/ijece.v14i3.pp3021-3035

Minh-Linh Dam
Posts and Telecommunications Institute of Technology
4 PUBLICATIONS **11** CITATIONS

Duy- Hung Ha
Ton Duc Thang University
62 PUBLICATIONS **400** CITATIONS

Han Minh Chau
HUTECH University
3 PUBLICATIONS **11** CITATIONS

Real-time phishing detection using deep learning methods by extensions

Dam Minh Linh[1], Ha Duy Hung[2], Han Minh Chau[3], Quang Sy Vu[4], Thanh-Nam Tran[5]
[1]Department of Information Systems, Faculty of Information Technology, Posts and Telecommunications Institute of Technology, Ho Chi Minh, Vietnam
[2]Wireless Communications Research Group, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[3]Department of Computer Networks, Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam
[4]Faculty of Automotive Engineering, School of Technology, Van Lang University, Ho Chi Minh City, Vietnam
[5]Data Science Laboratory, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received Oct 11, 2023
Revised Feb 21, 2024
Accepted Feb 25, 2024

Keywords:

Anti-phishing
Character embedding
Cybersecurity
Deep learning
Extension
Malicious URLs
Phishing detection

ABSTRACT

Phishing is an attack method that relies on a user's insufficient vigilance and understanding of the internet. For example, an attacker creates an online transaction website and tricks users into logging into the fake website to steal their personal information, such as credit card numbers, email addresses, phone numbers, and physical addresses. This paper proposes implementing an extension to prevent phishing for internet users. In particular, this study develops a smart warning feature for the proposed extension using deep learning models. The proposed extension installed in the web browser protects users by checking for, warning about, and preventing untrusted connections. This study evaluated and compared the performance of machine learning models using a malicious uniform resource locator (URL) dataset containing 651,191 data samples. The results of the investigation confirm that the proposed extension using a convolutional neural network (CNN) achieved a high accuracy of 98.4%.

Corresponding Author:

Ha Duy Hung
Wireless Communications Research Group, Faculty of Electrical and Electronics Engineering,
Ton Duc Thang University
Ho Chi Minh City, Vietnam
Email: haduyhung@tdtu.edu.vn

1. INTRODUCTION

Criminals committing internet fraud are increasing in number and professionalism. Cyberattacks are diverse, sophisticated, and widespread. Internet fraud often includes the theft of confidential information from an organization or individual for blackmail purposes, creating significant challenges for cybersecurity experts.

Recent studies have successfully detected phishing attacks on the internet. For example, Dhamdhere and Vanjale [1] focused on real-time phishing email classification using the k-means algorithm. The author used 160 emails from computer engineering students and divided them into three types: technical parameters, structure, and contents. The results obtained for true positives for legitimate and phishing were 67% and 80%, respectively, and true negatives were 30 % and 20%, respectively. In another study, Bahnseny *et al.* [2] used two algorithms, random forest (RF) and long short-term memory (LSTM), to evaluate the PhishTank and Common Crawl datasets, achieving accuracies of 93.5% and 98.7%, respectively. Similarly, Kumar *et al.* [3] evaluated using the MATLAB tool and a hybrid method supported by support vector machine (SVM) to detect phishing combined with feature extraction and message classification. The comparison results of the SVM models showed an accuracy of 87%, sensitivity of 88.5 %, and specificity of 91%, while the neural network model achieved an accuracy of 90.5%, sensitivity of 92%, and specificity of 93.5 %. However, the author's proposed model achieved the best results, with an accuracy of 98%, sensitivity of 97%, and specificity of 97.5%.

In study [4], features contributing to phishing susceptibility among students of petroleum resources in Nigeria were assessed. In this method, each participant's personal data were extracted from their profile to create content for appropriate emails containing compromised links that required immediate response. Students who clicked on the link or logged in to enter their details were scammed. The results showed that the school year progression reduced the number of phishing participants and email click-to-open rates. Increased computer time and network awareness correlated with lower click-through rates. Son and Dung [5] used a detection technique in anomaly detection applications based on two clustering techniques, k-means and density-based spatial clustering of application with noise (DBSCAN). Their method was evaluated using different feature extraction methods and clustering algorithms. The experimental results on the NSL-KDD dataset had a detection accuracy of over 97%. In other research, Salah and Zuhair [6] proposed predictive model was trained and tested on a dataset of 14,000 phishing uniform resource locators (URLs) and 28,074 legitimate URLs. The experiments' performance outputs were remarkable, with a 0.01% false positive rate and 99.27% testing accuracy.

Chanti *et al.* [7] combined two datasets, including two types of data labels, phishing and legitimate URLs, and extracted 19 out of 46 URL features from the datasets. The challenge introduced was to detect phishing URLs with valid secure sockets layer (SSL) certificates. To solve this problem, features related to digital certificates and URLs were combined to improve efficiency. In study [8], integrating the predictions of multiple LSTM models using the bagging and superposition methods was proposed. Different URL-based lexical features were extracted and classified using the LSTM method without performing separate feature extraction. The results were more accurate, with a low false positive rate of less than 0.15% performed on a large dataset. The dataset classification employed two algorithms, decision tree (DT) and RF [9], using the entropy feature combined with six features and achieving 96.30% accuracy.

Verma *et al.* [10] explained that email phishing has become a significant threat to all and is increasing daily. They provided a detailed description of the classification of phishing emails using natural language processing (NLP) concepts and calculated the accuracy rates of various classifiers. Khan *et al.* [11] focused on detecting malicious URLs using binary classification through the AdaBoost algorithm. The research team attempted an exact formulation of malicious URL exposure from a machine learning (ML) perspective and proposed using the AdaBoost algorithm, which yielded higher accuracy than other existing algorithms. Korkmaz *et al.* [12] proposed a hybrid phishing detection system using deep learning (DL)-based URL and content analysis. They conducted a study to create a large URL and content dataset by obtaining 36,173 phishing and 51,316 legitimate data records from PhishTan. With this new dataset, URL-based, content-based, and hybrid models were tested separately. Nordin *et al.* [13] proposed a comparative analysis of metaheuristic algorithms in fuzzy modeling for phishing attack detection. In both datasets, butterfly optimization algorithm (BOA) outperformed six other metaheuristic algorithms, including genetic algorithm, differential evolution, particle swarm optimization, teaching-learning-based optimization, harmony search, and gravitational search algorithm.

In study [14], ML technique for the phishing URL classification task achieved the highest accuracy of 98% with a naïve Bayes classifier. The malicious URLs dataset contained only about 57,000 to 60,000 phishing URLs; the rest were benign URLs. The dataset comprised mixed samples, with 70% used for training and 30% for testing. Yuan *et al.* [15] proposed a character embedding model that automated the vector representations of URLs, addressing the shortcomings of traditional phishing detection approaches that require manual features. Their dataset contained 587,668 phishing URLs and 584,909 legitimate URLs. The legitimate URLs were obtained from the top one million legitimate websites ranked by Alex5 on November 07, 2017, while the phishing URLs were downloaded from two data sources, PhishTank's open database1 and Reasonable Antiphishing7. The dataset was divided into 879,433 URLs used for training and 293,144 URLs used for testing. Their experiment used a DL model to achieve 99.79% accuracy.

Vijayalakshmi *et al.* [16] discussed research avenues for future investigation and proposed new ideas to develop the best online anti-phishing technique by illustrating phishing of the official PayPal website and a fraudulent PayPal website. Tang and Mahmoud [17] presented state-of-the-art solutions, compared and analyzed the challenges and limitations of each solution, and proposed research directions and future solutions. Their paper's main contributions included a presentation of the phishing life cycle that captures the phishing problem and a comprehensive analysis of ML-based approaches to detect phishing websites.

The ISCX URL-2016 dataset was utilized in the study [18], experimenting with 11,964 instances of legitimate and phishing URLs. The highest accuracy of 99.57% was achieved using the RF algorithm. Their proposed architecture for detecting phishing is as follows:

- Analyze the components of a URL in detail: protocol, subdomain, domain name, port, path, query, parameters, fragment.
  (e.g., *https://video.google.co.uk:80/videoplay?docid=7246927613731078230&hl=en#00h02m30s*).
- Chop a URL and perform lexical extraction to produce lexical data.
- In the ML classifier, if the result is legitimate, provide access is given; otherwise, deny access.

Aljofey *et al.* [19] used an effective phishing detection model based on character-level convolutional neural networks (CNNs) from a URL. The embedding layer was generally applied in the first layer of the CNN structure for an NLP problem using a tokenizer and vocabulary. The tokenizer processed URLs at the character level and added a unknown (UNK) token to the vocabulary. After fitting the training data, the tokenizer contained all the necessary information about the data and vocabulary. The alphabet consisted of 95 characters corresponding to 95 encoded numbers. These included the 26 letters of the English alphabet in lowercase letters (encoded numbers 1–26) and uppercase letters (encoded numbers 27–52), numbers 0–9 (encoded numbers 53–62), and 33 non-alphanumeric characters (encoded numbers 63–94). UNK corresponded to 95, and default corresponded to 0.

To obtain a solution to a large problem with high accuracy in identifying potentially fraudulent URL links on the internet in cybersecurity, ML algorithms are essential. In this study, we first develop a method to detect fraudulent URL links based on the characteristic components of a URL link combined with a full character set. Then, we feed a set of trained ML models to the malicious URL dataset, and finally, we design a prediction model to evaluate the results for benign URL links or phishing URL links. The main objectives of this research paper are to present proposals for new solutions and, based on the results of the studies in [18], [19], to inherit the dataset and improve features for the character set. While previous research has been limited, we make the following implementations and contributions:

- A new formula to optimize the ratio between training data and testing data. We use a variety of ML and DL models, including logistic regression (LR), DT, RF, SVM, CNN, and CNN-LSTM to evaluate the malicious URL dataset using mathematical formulas for properties such as accuracy, precision, and detection rate (i.e., recall). The experiment evaluates ML and DL models, combining large datasets scaled between training and testing in significant detail. Out of the six models trained on the dataset, the model with the highest accuracy is selected to detect malicious URL links.
- A comprehensive analysis and evaluation of new datasets, such as our malicious URL dataset (651,191 URLs), which contains a variety of large data sources with several fraudulent or high-risk URL links. The research article [18] contains a dataset of 11,964 instances of legitimate and phishing URLs that were inherited, reused, and now collected in the large malicious URL dataset.
- Improved efficiency in the accurate detection of fraudulent URL link data in the ML model in [19], which used a character set of 95 characters. We propose a character set character with 270 characters, which is used to train the input data of our ML model. The effectiveness of the new character set is much greater than the previous character set of 175 characters, which means an increased recognition of a phishing URL link. This is because a phishing URL link can use any of certain special characters, so the identification character set must be truly rich and diverse.
- Increased convenience and support for online web users and increased safety in online transactions. We propose and develop a software extension installed directly on web browsers with the Chromium kernel (e.g., Google Chrome and Microsoft Edge) to detect malicious URL links in real time using DL (CNN) techniques. We survey and apply a large malicious URL dataset (651,191 URLs) from [20] and [21], which were extracted from five benchmark datasets, [22]–[26], as the main data source for phishing detection websites.

The remainder of the paper is structured as follows: section 2 proposes a detection model, the extension software, and a processing algorithm combined with the DL model. Section 3 presents an ML model, training dataset, and evaluation method. The results are presented in section 4 with an evaluation and discussion of the research. Finally, section 5 concludes the study.

2. PROPOSED ALGORITHM

This section presents the proposed detection model, extension software, and the steps in the processing algorithm combined with the DL model, in particular, the detection model for identifying a URL as malicious or benign. The extension software, programmed using the JAVA programming tool, is installed directly on the web browser. The processing algorithm combined with the DL model includes computational techniques for detecting URL links as phishing or benign, and the CNN model was chosen as the main model because it has higher accuracy than models such as LR, DT, RF, SVM, and CNN-LSTM.

2.1. Proposed detection model

Figure 1 shows the steps for identifying a URL as malicious or benign, which are as follows:

a. The server trains the dataset of URL links using a DL algorithm.
b. An extension is developed and installed into the web browser at the client. The extension sends the URL to the server.
c. The server receives a URL from the client. The URL is classified using DL on the server.
d. If the URL is malicious, the client receives a warning message from the server.

The proposed model includes a software extension and a processing algorithm combined with the DL model.

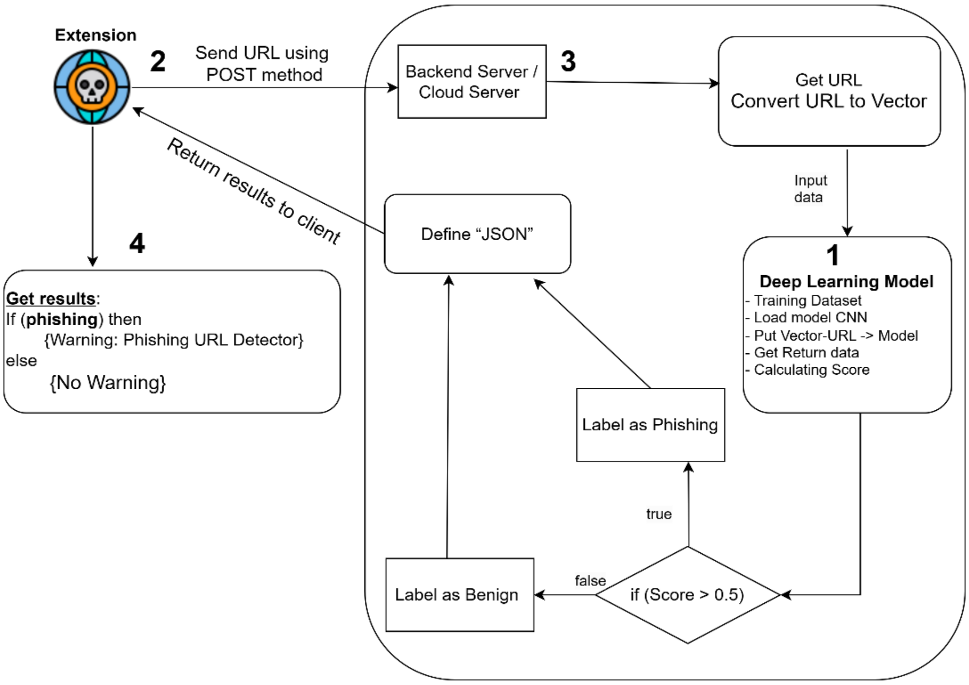


Figure 1. Architecture of the insecure link detection method

2.2. Proposal 1: Software extension

The two main tasks of the software extension are sending input data to the server and receiving results from the server. The extension is installed on the web browser with the Chromium kernel (e.g., Google Chrome and Microsoft Edge):

Send URL to the server (e.g., backend server or cloud server).

```
{
   - The user submits the URL link data to the web browser.
   - Call an application programming interface (API) address from the server
     (e.g., theUrl=http://127.0.0.1:5000/predict).
   - Send data in JSON format to the server (e.g., backend server or cloud server).
}
```

Receive URL results from the server with type RESPONSE.JSON

```
{if (obj.phishing==true) {Notification (Phishing Detected!!)}}.
```

2.3. Proposal 2: Processing algorithm combined with the DL model

The data are received from the extension, and the results are processed, calculated (e.g., steps 1–9), and then returned to the extension.

Backend server or cloud server:

{

Step 1: Use a CNN to train the dataset with input parameters

(e.g., *character_size=270*, i.e., the length of the encoding is marked as *{'UNK':1, 'e':2, 't':3, ..., 'ž':268, '‰':269, 'ത':270}*; embedding *size=128*; input length is the length of data to be trained and tested).

```
Step 2: Receive URL data in JSON format.
Step 3: Load the tokenizer term encryption function (the encryption function was packaged previously).
Step 4: Convert the data type between URL and tokenizer to vectorizer
        (e.g., vectorize_url(tokenizer,url)).
Step 5: Load the DL model previously trained on the dataset.
Step 6: Classify the URL and make the prediction.
Step 7: The URL link is separated word by word using the tokenizer character dictionary. Then, the data are converted to a vector and
        fed into the trained model to predict the results.
        (e.g., Score=predict1(model, vectorizer, url)).
Step 8: Process prediction probability:

        If (Score > 0.5)
         Label:{phishing}
         else Label:{benign}

Step 9: Return the data to the extension in JSON format.
}
```

Classification thresholds (i.e., Score, in our case) applied in ML involve real time [27]. Set the threshold to determine which specific label to assign to each prediction. A threshold classification metric of 0.5 is usually the default choice, but it can range from 0 to 1. Here, if Score is greater than 0.5, the URL link is labeled as phishing; otherwise, the URL link is labeled as benign.

As shown in Figure 2, our alphabet consists of 270 characters corresponding to 270 encoded numbers. These include the 26 letters of the English alphabet in lowercase letters, numbers 0–9, and 234 non-alphanumeric characters, where the length of the encoding is marked as *{‘UNK’:1, ‘e’:2, ‘t’:3, ..., ‘ž’:268, ‘‰’:269, ‘๓’:270}*.

{'UNK':1, 'e':2, 't':3, 'o':4, 'a':5, 'i':6, '/':7, 'n':8, 'c':9, 's':10, 'r':11, 'm':12, '.':13, 'l':14, 'p':15, 'd':16, 'h':17, '-':18, 'w':19, 'u':20, 'b':21,
'g':22, '1':23, 'f':24, '0':25, '2':26, 'y':27, '=':28, 'k':29, '3':30, 'v':31, '8':32, '%':33, '5':34, '4':35, '9':36, '_':37, '7':38, '6':39, '&':40, 'x':41,
':':42, 'j':43, 'z':44, '?':45, 'q':46, '+':47, ';':48, '\\':49, “"“:50, '~':51, '(':52, ')':53, 'ะ':54, ', ':55, ']':56, '[':57, '@':58, '!':59, 'พ':60, 'ต':61,
'#':62, '|':63, '':64, 'ฝ':65, 'ธ':66, 'ฐ':67, 'ร':68, 'â':69, '*':70, 'é':71, 'ผ':72, 'ã':73, 'บ':74, '\x82':75, '\x83':76, 'ฒ':77, '}':78, '{':79, '$':80,
'ฑ':81, 'ณ':82, 'ฟ':83, 'ฮ':84, 'ป':85, '^':86, 'ò':87, 'ô':88, 'è':89, 'ñ':90, 'ด':91, 'õ':92, 'น':93, 'ì':94, 'ฆ':95, 'æ':96, 'î':97, 'ù':98, 'ö':99,
'ê':100, 'å':101, 'ë':102, '\xa0':103, 'ก':104, 'ó':105, 'ç':106, 'ü':107, 'ø':108, 'ú':109, 'ý':110, 'ι':111, 'í':112, 'ä':113, 'ï':114, 'á':115, 'à':116,
'û':117, '>':118, '\x06':119, 'ย':120, 'ð':121, 'þ':122, 'ท':123, '<':124, '`':125, '\x0f':126, '°':127, '·':128, '\x89':129, '¹':130, '½':131,
'\x88':132, '»':133, 'º':134, '¿':135, '¡':136, '³':137, '\x96':138, '\x8d':139, '\x9b':140, '\x9a':141, '¬':142, '©':143, '¶':144, '\x02':145,
'¯':146, '\x87':147, '\x11':148, '\x04':149, '\x12':150, '®':151, '\x97':152, '\x10':153, '\x13':154, 'ถ':155, '\x17':156, '¦':157, 'ง':158,
'\x9e':159, '\x1b':160, '¤':161, '\x1c':162, '¸':163, 'µ':164, '¾':165, 'ญ':166, '\x93':167, '\x08':168, '\x85':169, '£':170, '×':171, 'ฉ':172,
'า':173, 'ß':174, '÷':175, '\x15':176, '“':177, '\x8e':178, '±':179, '\x8c':180, '§':181, '´':182, '¨':183, '\x84':184, '\x18':185, '\x1f':186,
'\x91':187, '¢':188, 'ล':189, '\x92':190, '\x0e':191, '\x0c':192, '\x95':193, '\x1a':194, '¥':195, '\x07':196, '\x81':197, 'ค':198, '¼':199,
'\x98':200, '\x9c':201, '\x0b':202, '\x99':203, '\x94':204, '²':205, '\x19':206, 'ª':207, '\x01':208, '\xad':209, '\x7f':210, '\x1e':211,
'\x90':212, '\t':213, '\x9d':214, '\x14':215, '\x80':216, '\x86':217, '\x8b':218, '\x03':219, '«':220, '\x1d':221, '–':222, '\x05':223, '\x8a':224,
'\x16':225, 'ฦ':226, 'ช':227, '\x9f':228, 'ฅ':229, 'ÿ':230, '\x8f':231, '”':232, 'ห':233, 'ข':234, 'ซ':235, 'จ':236, 'ฎ':237, 'ฤ':238, 'ี':239,
'š':240, 'ฏ':241, 'ั':242, '\n':243, '\r':244, 'ึ':245, '‚':246, 'ๅ':247, 'ส':248, '“':249, 'ศ':250, 'ฌ':251, '“':252, '—':253, '�':254, 'ไ':255,
'ำ':256, 'ุ':257, 'ฬ':258, 'ƒ':259, 'ํ':260, '•':261, 'ฺ':262, '抛':263, '傳':264, 'ว':265, 'ิ':266, 'ี':267, 'ž':268, '‰':269, '๓':270}

Figure 2. Character vocabulary index representation.

3. CNN, LSTM, and CNN-LSTM models

3.1. CNN and CNN-LSTM models

CNN has three main layers: convolutional, pooling, and fully connected (FC) layers. The convolutional layer, the first layer, is used to extract the various features from the input data. After extracting features, the convolutional layer outputs the result to the next layer. The pooling layer, or down sampling layer, reduces the dimensionality of the input using average pooling and max pooling functions. The FC layer is usually positioned before the output layer in CNN architecture. The input data from the previous layers are flattened before passing to the FC layer. Consisting of weights and biases, the FC layer is used to fully connect the neurons between two different layers.

CNN has additional parameters, including the dropout layer and the activation function, as shown in Figures 3 and 4 for the CNN and CNN-LSTM models. In the dropout layer, the features are connected to the FC layer. Dropout improves the performance of an ML model as it drops neurons from the neural networks during training. The activation function is the CNN model’s most important parameter. Here, the activation function uses the Softmax function. Finally, the dense layer yields the results of the model’s output data after training the dataset. In the CNN model in Figure 3, the embedding layer is the model’s input layer using three parameters, such as *character_size=270*, which means the length of the encoding is marked as *{‘UNK’:1, ‘e’:2, ‘t’:3, ..., ‘ž’:268, ‘‰’:269, ‘ග’:270}*; *embedding size=128*; and input length, which is the length of data to be trained and tested. Dense is the model’s output layer evaluating results for two types of data: benign and phishing.

The combined CNN-LSTM model in Figure 4 shows that the embedding layer uses the same three main parameters as the proposed CNN model. The difference is that the LSTM model is inserted before the first dense layer, followed by the dropout layer, and finally, the output of the data results (second dense layer). After training the dataset for the CNN-LSTM combined model, the total number of parameters is 6,890,370. We simulated the values of layers in the algorithm models, as in [28].

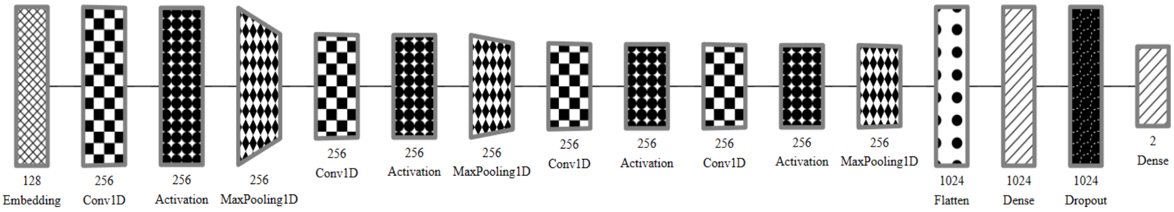


Figure 3. Proposed CNN architecture

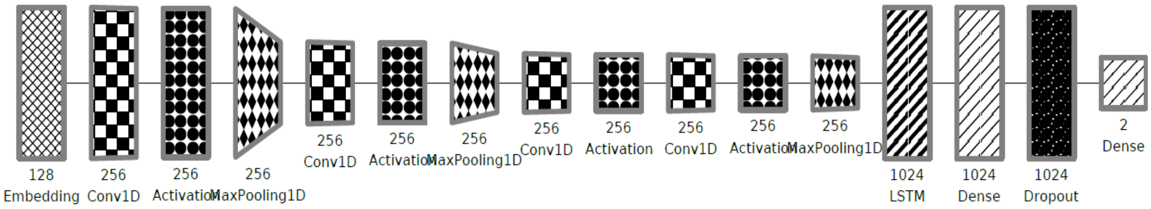


Figure 4. Proposed CNN-LSTM architecture

3.2. LSTM

As shown in Figure 5, the LSTM network architecture consists of three parts, as referenced in [29]–[32]. LSTM is an improved version of the recurrent neural network (RNN). The limitation of the RNN is short-term memory, i.e., the RNN only learns nearby states. Therefore, LSTM has been studied to solve this problem. In Figure 5, f_t, i_t, and o_t correspond to the forget, input, and output gates, respectively, such that:

$$f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i \tag{2}$$

$$o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o) \tag{3}$$

$$\tilde{c}_t = \tanh(U_c * x_t + W_c * h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{5}$$

$$h_t = o_t * \tanh(c_t) \tag{6}$$

where $0 < f_t, i_t, o_t < 1$ and b_f, b_i, b_o are bias coefficients in (1), (2), and (3), respectively. Next, in (4), a tanh function creates a vector of new candidate values, $\tilde{c}_t$, which can be added to the state. In (5), the forget gate decides the number from the previous cell state, and the input gate chooses the number from the input of the state and the hidden layer of the previous layer. In (6), the output gate decides how much to take from the cell state to become the output of the hidden state.

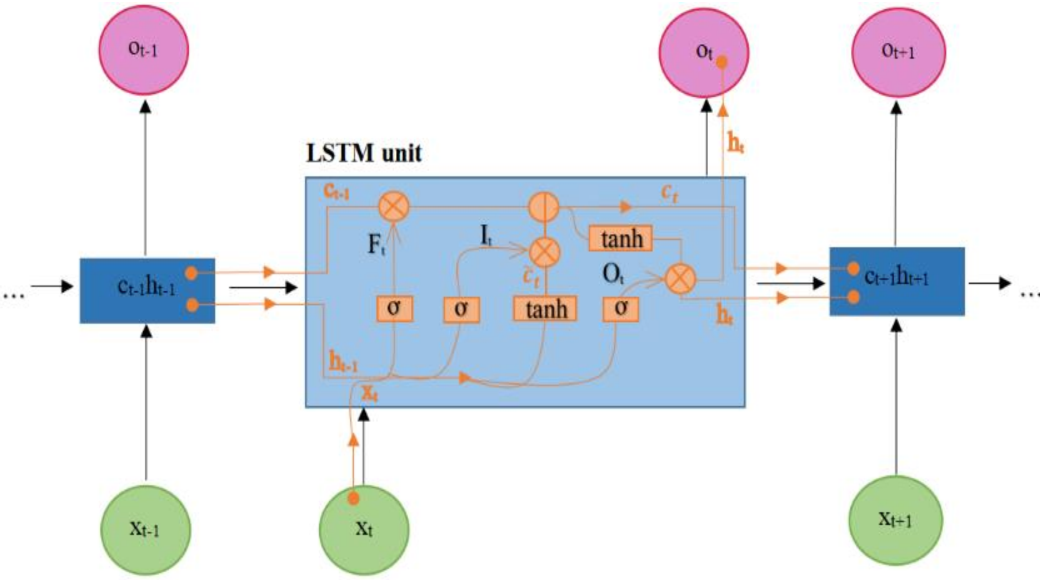


Figure 5. Architecture of the LSTM cell

3.3. Evaluation method

In this study, we use evaluation metrics, such as a confusion matrix (CM), accuracy, detection rate (*DR*) or recall, precision (*Pr*), and F1-score (*F1S*) to calculate the effectiveness of the training data in ML algorithms, referenced from [33]–[36]. The CM, as given in (7), is used to evaluate the accuracy of the dataset. The CM contains the predicted and actual classifiers, which are empirically evaluated on a dataset:

$$CM = [TN\ FP\ FN\ TP] \tag{7}$$

where *TP* represents true positives, i.e., correct malicious URLs predicted; *TN* represents true negatives, i.e., correct benign URLs predicted; *FP* represents false positives, i.e., incorrect malicious URLs predicted; and *FN* represents false negatives, i.e., benign URLs predicted. The other metrics are defined as in (8)–(11):

$$Ac = \frac{TP+TN}{TP + TN + FP + FN} \tag{8}$$

$$Pr = \frac{TP}{TP + FP} \tag{9}$$

$$F1S = \frac{2 * Pr * DR}{Pr + DR} \tag{10}$$

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

where *Ac* is the ratio of correct predictions to the total number of samples and *Pr* is defined as the number of true positives divided by the number of true positives plus the number of false positives. When *Pr*=1, the numerator and denominator are equal (*TP*=*TP* +*FP*) since *FP*=0. If *FP* increases, then the denominator is larger than the numerator, and the precision decreases. The F1-score (*F1S*) is the average of *Pr* and *DR*. The value is large if both *Pr* and *DR* are large. If *DR* is approximately 1, the classification is good.

3.4. Dataset training

Malicious URLs or malicious websites are serious cybersecurity threats. Cybersecurity researchers have collected this dataset, which includes numerous examples of malicious URLs, and then used an ML model to evaluate, identify, and prevent viruses on computer systems and reduce the risk to users. In studies [20] and [21], cybersecurity researchers collected a massive dataset of 651,191 URLs, as shown in Table 1. This dataset was selected and inherited from five sources: [22]–[26].

Table 1. Malicious URLs dataset

URL type	No. of URLs
Benign	428103
Defacement	96457
Phishing	94111
Malware	32520
Total	651191

4. RESULTS AND DISCUSSION

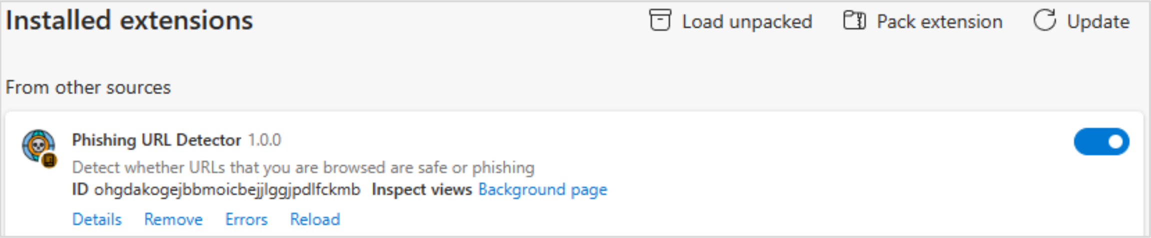
Experiments were conducted using a server with the following hardware configuration: two 2.30 GHz E5 2696 v3 CPUs (36 cores, 72 threads), RAM 64 Gb DDR4, NVIDIA® GeForce RTX™ 3060 GPU 12 Gb. The research team achieved results based on evaluation criteria, including accuracy, precision, and recall for LR, DT, RF, SVM, CNN, and CNN-LSTM, after training the dataset. The CNN model with the highest accuracy was selected as the detection model for the malicious data type. This CNN model was saved as a file and uploaded to an address, such as a backend or cloud server, to connect to the extension installed on the user's web browser. The goal was to check the safety of the user's URL links when logging into a web browser.

4.1. Extension setup

We used the JavaScript programming language to write the program with an extension called *pop-up.js*. This extension is installed on web browsers with Chromium kernel using the following steps:

```
Step 1: Obtain URL link on browser and send it to the backend or cloud server
Step 2: Check if the URL is in the whitelist
  {
    if(url!=whitelist)
    - Call to backend server using "POST" method
    - The attached data are the URL formatted to JSON
    - If the response is returned {if(obj.phishing==true)}
    {Display a warning message to the user}
  }
Step 3: Record the tab's runtime on the browser.
```

This extension was installed on web browsers, such as Google Chrome and Microsoft Edge, using version 1.0.0, as shown in Figure 6. Here, internet users can use a web browser on their personal computer to log on to any URL links. The URL links chosen depend on their purpose, as shown in Figure 7. A function quickly checks the trustworthiness of the URL link, as shown in Figure 8. This feature helps users by quickly checking the URL link. The usefulness of this extension is based on the very high safety efficiency for internet users in today's network environment. Its effectiveness is proven based on scientific and practical evaluation methods. ML models are used to accurately evaluate datasets, and the Java programming language is used to write the extension software program installed on the web browser.



Phishing URL Detector 1.0.0

Detect whether URLs that you are browsed are safe or phishing

ID ohgdakogejbbmoicbejjlggjpdlfckmb Inspect views Background page

Figure 6. Setup for browser extensions

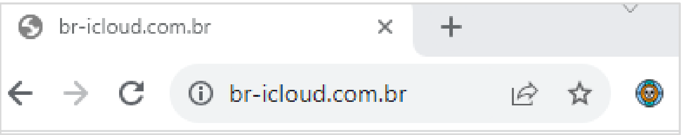


Figure 7. URL link accessed in a browser

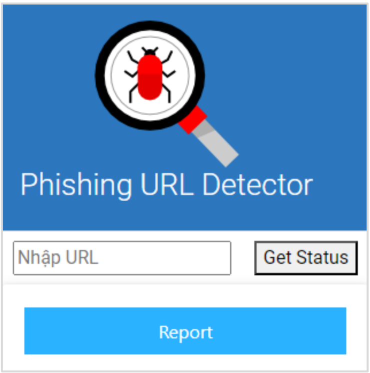
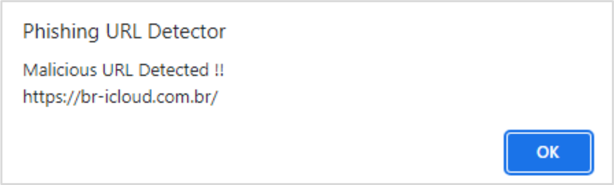


Figure 8. URL link reliability quick-check feature

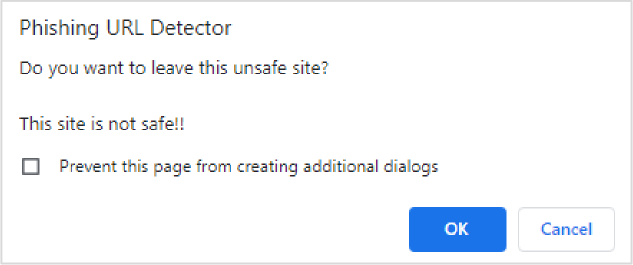
4.2. Practical experiment of the proposed extension

We employed a dataset containing malicious URLs (e.g., *http://br-icloud.com.br/*) and benign URLs (e.g., *https://www.google.com*) for the practical experiment. The dataset was first used for training the DL algorithm. After training, we input each URL into the web browser and checked the result. For example, we input a URL with *http://br-icloud.com.br/* into the web browser. Immediately, the proposed extension classified this URL with a phishing label and displayed a pop-up warning to users, as shown in Figure 9. The extension recommends users leave and block this site for safety, as shown in Figure 10.


Phishing URL Detector

Malicious URL Detected !!
https://br-icloud.com.br/

Figure 9. Warning that the URL link is phishing


Phishing URL Detector

Do you want to leave this unsafe site?

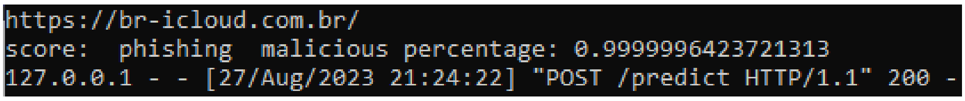
This site is not safe!!

☐ Prevent this page from creating additional dialogs

Figure 10. Recommendations and support for users

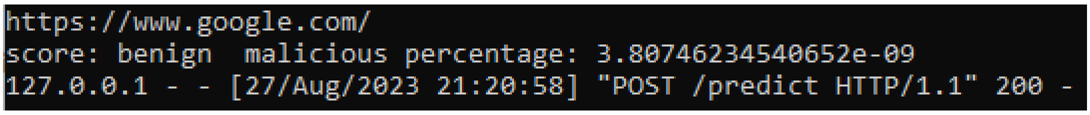
The server system issues warning information about the user's login process on the web browser. A phishing URL link (*http://br-icloud.com.br*) is detected, and a warning is given at a specific time (e.g., 27/Aug/2023 21:24:22), and a benign URL link (*https://www.google.com*) is alerted at a specific time (e.g., 27/Aug/2023 21:20:58), as shown in Figures 11 and 12, respectively. The real-time response efficiency of applying ML models is used to evaluate the accuracy of large datasets and predict or classify data. For example, an internet user enters a URL link with *http://br-icloud.com.br/* into the web browser. This malicious URL link was quickly detected.

Additionally, we observe the results in the second row in Figures 11 and 12. Figure 11 shows a URL link (e.g., *http://br-icloud.com.br*) that is classified as phishing since it has a score with a classification metric of approximately 0.999, which is greater than 0.5. Otherwise, a URL link (e.g., *https://www.google.com*) that is classified and labeled as benign will have a score with a classification metric of less than 0.5. Figure 12 shows that the classification metric is about 3.807e−09, and thus, the link is considered benign.



```
https://br-icloud.com.br/
score:  phishing  malicious percentage: 0.9999996423721313
127.0.0.1 - - [27/Aug/2023 21:24:22] "POST /predict HTTP/1.1" 200 -
```

Figure 11. Warning from the server about phishing URL link



```
https://www.google.com/
score: benign  malicious percentage: 3.80746234540652e-09
127.0.0.1 - - [27/Aug/2023 21:20:58] "POST /predict HTTP/1.1" 200 -
```

Figure 12. Notification from the server about benign URL link

4.3. Simulation and analysis results of algorithms

Equation (12) is used to create various training–testing ratios for evaluating the criteria of accuracy, precision, and recall for the LR, DT, RF, SVM, CNN, and CNN-LSTM models. For example, the test size (*Testing*) is the fraction of samples used for testing models. The training size (Training) is the fraction of samples used to train the models, i.e., the remainder of the total dataset. We define the ratio between these in Table 2. We use this formula to define the simulation scenarios so we can determine the optimal ratio for training and testing data to use for a predictive model.

$$Testing\ (i) = 1 - Training\ (i) \qquad (12)$$

where *i* is the simulation, i.e., 1–9.

The malicious dataset with 651,191 URLs is divided between training and testing data, as shown in Table 2. The training fraction column corresponds to the fraction of URL data samples used for training. The testing fraction column corresponds to the fraction of URL data samples used for testing. In the number of URL links, the training number gradually increases, and the testing number gradually decreases.

Table 2. Ratios of training and testing data in the dataset containing 651,191 URLs

Simulation iteration *i*	Training fraction	Number of training URLs	Testing fraction	Number of testing URLs	Training–testing ratio
1	0.1	65119	0.9	586072	1:9
2	0.2	130238	0.8	520953	2:8
3	0.3	195357	0.7	455834	3:7
4	0.4	260476	0.6	390715	4:6
5	0.5	325596	0.5	325595	5:5
6	0.6	390715	0.4	260476	6:4
7	0.7	455834	0.3	195357	7:3
8	0.8	520953	0.2	130238	8:2
9	0.9	586072	0.1	65119	9:1

Figure 13 and Table 3 compare the accuracy of the six algorithms on the training and testing datasets. That is, the data used for training are training fractions of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9, corresponding to the fraction used for testing of 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1. The CNN had the highest accuracy, while LR had the lowest. At the Training and Testing ratios of 1:9, algorithms from lowest to highest accuracy were as follows: LR, CNN-LSTM, DT, RF, SVM, and CNN. The accuracy of the LR algorithm increased by a negligible amount. At a training and testing ratio of 3:7, CNN overtook CNN-LSTM with the highest accuracy. For training and testing ratios ranging from 4:6 to 8:2, CNN's accuracy increased significantly, peaking at 98.416%. In contrast, CNN-LSTM dropped to only 94.4394% at a training fraction of 0.7 and testing fraction of 0.3 and remained almost unchanged for the rest of the range. The accuracies of DT and RF remained relatively stable at around 95% to 96%.

The line graph in Figure 13 shows the accuracy of ML and DL models in the training and testing datasets for the various testing and training ratios shown in Table 3. According to the results, the accuracy of the LR, DT, RF, SVM, CNN, and CNN-LSTM models are shown in this chart. The CNN model has the highest accuracy. Meanwhile, the remaining models are less effective than the CNN model in terms of accuracy.

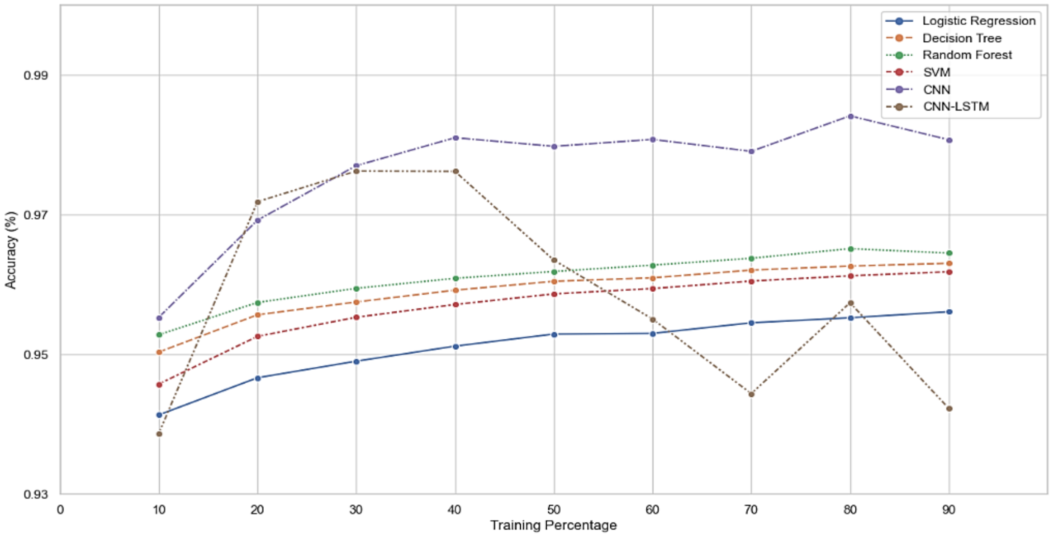


Figure 13. Line graph of ML and DL models' accuracy for each training and testing

Table 3. Accuracies of the six algorithms after training and testing on the dataset

Training–testing ratio	LR	DT	RF	SVM	CNN	CNN-LSTM
1:9	0.941333	0.950329	0.952811	0.945752	0.955296	0.938645
2:8	0.946659	0.955675	0.957441	0.952583	0.969222	0.971863
3:7	0.949034	0.957511	0.959466	0.955321	0.977022	0.976255
4:6	0.951187	0.959203	0.960902	0.957148	0.98103	0.976213
5:5	0.952917	0.960472	0.961879	0.958667	0.979797	0.963516
6:4	0.953013	0.960972	0.962788	0.959436	0.980785	0.955071
7:3	0.95453	0.962075	0.963769	0.960508	0.979079	0.944394
8:2	0.955251	0.962653	0.965149	0.961256	0.98416	0.957398
9:1	0.956112	0.963053	0.964512	0.96184	0.980743	0.942276

Table 4 compares the precision among the six algorithms in nine training–testing ratios. The precision of five algorithms (CNN, SVM, RF, DT, and LR) generally increased with the number of training data for all scenarios, while CNN-LSTM's precision reduced. For example, at a ratio of 9:1, CNN's precision increased from 95.5296% to 98.403%. Similarly, at this ratio, SVM's precision slightly increased from 97.0865% to 97.8425%, and those of RF, DT, and LR grew by 1.6823, 1.4005, and 1.6868 percentage points, respectively. The trend for the CNN-LSTM algorithm was noticeably different, where its precision significantly declined from 97.6255% at 3:7 to only 94.2276% at 9:1.

Table 4. Precision of the algorithms after training and testing on the dataset

Training–testing ratio	LR	DT	RF	SVM	CNN	CNN-LSTM
1:9	0.953395	0.955295	0.960274	0.970865	0.955296	0.938645
2:8	0.959935	0.962721	0.968009	0.973721	0.969222	0.971863
3:7	0.962764	0.964717	0.97134	0.974552	0.977022	0.976255
4:6	0.965149	0.966661	0.973789	0.976189	0.98103	0.976213
5:5	0.967276	0.967118	0.972877	0.976701	0.979797	0.963516
6:4	0.967458	0.966566	0.972789	0.977598	0.980785	0.955071
7:3	0.968944	0.967997	0.976255	0.978427	0.979079	0.944394
8:2	0.967958	0.968841	0.976552	0.978547	0.98403	0.957398
9:1	0.970263	0.9693	0.977097	0.978425	0.980743	0.942276

Table 5 shows that CNN's recall value was highest, and LR's recall was lowest during the simulations. At the training–testing ratio of 1:9, LR reached 87.1346%, while DT, RF, SVM, and CNN-LSTM achieved 89.6986%, 89.9467%, 86.769%, and 93.8645% recall, respectively. From a ratio of 7:3 to 3:7, CNN overtook CNN-LSTM with the highest recall. From 4:6 to 9:1, CNN's recall increased steadily, peaking at 98.425%. In contrast, CNN-LSTM's recall declined to only 94.4394% at 7:3 and remained almost unchanged for the rest of the range. The recalls of DT and RF stayed relatively stable at around 92% and 91%, respectively.

Table 5. Evaluation of the recall of six algorithms after training and testing on the dataset

Training–testing ratio	LR	DT	RF	SVM	CNN	CNN-LSTM
1:9	0.871346	0.896986	0.899467	0.86769	0.955296	0.938645
2:8	0.881072	0.905687	0.905704	0.885488	0.969222	0.971863
3:7	0.885478	0.909229	0.908486	0.8929	0.977022	0.976255
4:6	0.88964	0.912381	0.910379	0.89679	0.98103	0.976213
5:5	0.892769	0.915755	0.914213	0.900837	0.979797	0.963516
6:4	0.892878	0.917824	0.917028	0.902269	0.980785	0.955071
7:3	0.895991	0.919704	0.916536	0.904672	0.979079	0.944394
8:2	0.899144	0.920593	0.920368	0.906786	0.98425	0.957398
9:1	0.899458	0.921332	0.917926	0.908647	0.980743	0.942276

4.4. Discussion

In recent years, cybersecurity has been a challenging topic for many scientific researchers and even cybersecurity experts. Many research projects have been published with various solutions related to protecting users from the risks of phishing attacks. Advanced technologies such as artificial intelligence (AI), ML, and DL combine methods of analyzing large cybersecurity-relevant datasets, such as NLP and clustering algorithm techniques. Regardless of the approach, the solution must be validated using mathematical methods and accurate results. Therefore, our research contributes new and improved proposals to previous studies, which have many limitations. We provide professionalism when deploying software in the form of an extension installed directly on the web browser, whose main task is to check URL links the user accesses. We divided a large malicious URL dataset (651,191 URLs) into training and testing datasets of various ratios to evaluate the accuracy of six models: LR, DT, RF, SVM, CNN, and CNN-LSTM, using formulas for accuracy, precision, and recall. After training the six ML models, the DL CNN model, which had the highest accuracy of approximately 98.5% at the optimal of ratio of 8:2, was chosen as the detection and prediction model for data in the form of benign or phishing URL links. These recommendations provide effective convenience and protect users from the dangers of internet fraud.

The improvement in the character set diversity of 270 characters replaces the 95 characters in the previous study, with the goal of increasing recognition of a phishing URL link. Because the structure of a phishing URL link often uses many special characters to form a URL link, using a character set with many characters is essential for the input data of the ML model. The large malicious URL dataset has numerous phishing URL links gathered from five data sources, including the dataset mentioned in the previous study, which has about 11,964 instances of legitimate and phishing URLs. The more phishing URL links in the dataset and the higher the accuracy of the training model, the more effective the model is at detecting phishing URL links. The detection technique is based on extracting features from the character set and analyzing the components of a standard URL link, processing data in vectorizer form, and predicting results for the data.

To increase support for online users and improve safety in online transactions, we propose our software extension installed directly on web browsers with the Chromium kernel (e.g., Google Chrome and Microsoft Edge) to detect malicious URL links in real time using DL CNN techniques. We validated the software with the large malicious URL dataset as the main data source for phishing detection websites, which is the scientific basis for the research conducted in this study, while previous studies are still limited.

A comprehensive refresh of datasets related to the field of cybersecurity since most of the datasets have been published for a long time. Among them, many URL links no longer exist, and some URL links that still exist include data labels that are both phishing and benign, making phishing URL links very difficult to detect and evaluate. As a result, an appropriate policy or recommendation is difficult to define for internet users.

5. CONCLUSION

In this paper, a malicious URL detection method using an extension was designed to help internet users identify whether a URL is legitimate or malicious. The extension was evaluated using six ML and DL algorithms that were trained and tested on the dataset using the CM method with evaluation criteria of accuracy, prediction, and recall. The CNN algorithm overtook the remaining algorithms, LR, DT, RF, SVM, and CNN-LSTM, with the highest accuracy of 98.4% at a ratio of 8:2 is best for training and testing data, respectively. The new approach uses a character set of 270 characters to evaluate ML and DL models effectively.

In the near future, the research team will introduce this convenience, provide a trial version of this extension software to online forums and communities, and install the DL CNN model on the cloud server. Hopefully, with completely free support, it will be truly useful for internet users, and we will publish practical scientific research in the field of cybersecurity.

ACKNOWLEDGMENTS

The research leading to these results was financially supported by the Posts and Telecommunications Institute of Technology under the Ministry of Information and Communications (MIC).

REFERENCES

[1] V. Mhaske-Dhamdhere and S. Vanjale, "A novel approach for phishing emails real time classification using k-means algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 5326–5332, 2018, doi: 10.11591/ijece.v8i6.pp.5326-5332.

[2] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonzalez, "Classifying phishing URLs using recurrent neural networks," *eCrime Researchers Summit*, pp. 1–8, 2017, doi: 10.1109/ECRIME.2017.7945048.

[3] A. Kumar, J. M. Chatterjee, and V. G. Díaz, "A novel hybrid approach of SVM combined with NLP and probabilistic neural network for email phishing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 486–493, 2020, doi: 10.11591/ijece.v10i1.pp486-493.

[4] R. E. Yoro, F. O. Aghware, B. O. Malasowe, O. Nwankwo, and A. A. Ojugo, "Assessing contributor features to phishing susceptibility amongst students of petroleum resources varsity in Nigeria," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 2, pp. 1922–1931, 2023, doi: 10.11591/ijece.v13i2.pp1922-1931.

[5] N. H. Son and H. T. Dung, "A lightweight method for detecting cyber attacks in high-traffic large networks based on clustering techniques," *International Journal of Computer Networks and Communications*, vol. 15, no. 1, pp. 35–51, 2023, doi: 10.5121/ijcnc.2023.15103.

[6] H. Salah and H. Zuhair, "Deep learning in phishing mitigation: a uniform resource locator-based predictive model," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 3, pp. 3227–3243, 2023, doi: 10.11591/ijece.v13i3.pp3227-3243.

[7] S. Chanti, T. Chithralekha, and K. S. Kuppusamy, "PUMPP: phishing URL detection using machine learning with monomorphic and polymorphic treatment of features," *International Journal of Computer Networks and Communications*, vol. 15, no. 3, pp. 93–112, 2023, doi: 10.5121/ijcnc.2023.15306.

[8] B. Banik and A. Sarma, "Phishing URL detection using LSTM based ensemble learning approaches," *International Journal of Computer Networks and Communications*, vol. 15, no. 1, pp. 17–33, 2023, doi: 10.5121/ijcnc.2023.15102.

[9] T. Chandrasegar and P. Viswanathan, "Dimensionality reduction of a phishing attack using decision tree classifier," in *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, Mar. 2019, pp. 1–4, doi: 10.1109/i-PACT44901.2019.8960117.

[10] P. Verma, A. Goyal, and Y. Gigras, "Email phishing: text classification using natural language processing," *Computer Science and Information Technologies*, vol. 1, no. 1, pp. 1–12, 2020, doi: 10.11591/csit.v1i1.p1-12.

[11] F. Khan, J. Ahamed, S. Kadry, and L. K. Ramasamy, "Detecting malicious URLs using binary classification through AdaBoost algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 997–1005, 2020, doi: 10.11591/ijece.v10i1.pp997-1005.

[12] M. Korkmaz, E. Kocyigit, O. K. Sahingoz, and B. Diri, "A hybrid phishing detection system using deep learning-based URL and content analysis," *Elektronika ir Elektrotechnika*, vol. 28, no. 5, pp. 80–89, 2022, doi: 10.5755/j02.eie.31197.

[13] N. S. Nordin *et al.*, "A comparative analysis of metaheuristic algorithms in fuzzy modelling for phishing attack detection," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 23, no. 2, pp. 1146–1158, Aug. 2021, doi: 10.11591/ijeecs.v23.i2.pp1146-1158.

[14] J. Kumar, A. Santhanavijayan, B. Janet, B. Rajendran, and B. S. Bindhumadhava, "Phishing website classification and detection using machine learning," in *2020 International Conference on Computer Communication and Informatics (ICCCI)*, Jan. 2020, pp. 1–6, doi: 10.1109/ICCCI48352.2020.9104161.

[15] H. Yuan, Z. Yang, X. Chen, Y. Li, and W. Liu, "URL2Vec: URL modeling with character embeddings for fast and accurate phishing website detection," in *2018 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Ubiquitous Computing and Communications, Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications (ISPA/IUCC/BDCloud/SocialCom/Sus*, Dec. 2018, pp. 265–272, doi: 10.1109/BDCloud.2018.00050.

[16] M. Vijayalakshmi, S. Mercy Shalinie, M. H. Yang, and U. Raja Meenakshi, "Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions," *IET Networks*, vol. 9, no. 5, pp. 235–246, 2020, doi: 10.1049/iet-net.2020.0078.

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

[18] 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," *Computer Communications*, vol. 175, pp. 47–57, 2021, doi: 10.1016/j.comcom.2021.04.023.

[19] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J.-P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, Sep. 2020, doi: 10.3390/electronics9091514.

[20] M. Siddhartha, "Malicious URLs dataset," *Kaggle*. https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset (accessed May 12, 2022).

[21] D. M. Linh, "Malicious_651191URLs_ManuSiddhartha," *GitHub*. https://github.com/MinhLinhEdu/Malicious_651191URLs_ManuSiddhartha/releases/tag/phishing (accessed Sep 24, 2022).

[22] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious URLs using lexical analysis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9955, 2016, pp. 467–482.

[23] J. S. B. Branner, "RiskAnalytics," *GitHub*. https://github.com/topics/malicious-domains (accessed Feb. 18, 2023).

[24] F. Ahmad, "Using machine learning to detect malicious URLs," *GitHub*. https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs/tree/master/data (accessed Oct. 15, 2022).

[25] C. T. I. G. (Talos), "Phishtank," *Cisco Talos Intelligence Group*. https://www.phishtank.com/developer_info.php (accessed Feb. 22, 2021).

[26] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: detecting phishing with streaming analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, Dec. 2014, doi: 10.1109/TNSM.2014.2377295.

[27] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, "Finding the best classification threshold in imbalanced classification," *Big Data Research*, vol. 5, pp. 2–8, 2016, doi: 10.1016/j.bdr.2015.12.001.

[28] D. M. Linh, “Viscom.” https://viscom.net2vis.uni-ulm.de/bnAPyb6HFRP4a0bJgi4xqmCM9C1TEl2SREH7Uju4T8kSFkQc2z (accessed Jun. 15, 2023).

[29] T. T. Huynh and H. T. Nguyen, “On the performance of intrusion detection systems with hidden multilayer neural network using DSDtraining,” *International Journal of Computer Networks and Communications*, vol. 14, no. 1, pp. 117–137, 2022, doi: 10.5121/ijcnc.2022.14108.

[30] A. Muneer, R. F. Ali, A. Almaghthawi, S. M. Taib, A. Alghamdi, and E. A. A. Ghaleb, “Short term residential load forecasting using long short-term memory recurrent neural network,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 5, pp. 5589–5599, 2022, doi: 10.11591/ijece.v12i5.pp5589-5599.

[31] M. Sher, N. Minallah, T. Ahmad, and W. Khan, “Hyperparameters analysis of long short-term memory architecture for crop classification,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 4, pp. 4661–4670, 2023, doi: 10.11591/ijece.v13i4.pp4661-4670.

[32] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, “Deep learning with convolutional neural network and long short-term memory for phishing detection,” in *2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, Aug. 2019, pp. 1–8, doi: 10.1109/SKIMA47702.2019.8982427.

[33] S. Al-Ahmadi and Y. Alharbi, “A deep learning technique for web phishing detection combined url features and visual similarity,” *International Journal of Computer Networks and Communications*, vol. 12, no. 5, pp. 41–54, 2020, doi: 10.5121/ijcnc.2020.12503.

[34] K. Roshan and A. Zafar, “Utilizing xai technique to improve autoencoder based model for computer network anomaly detection with Shapley additive explanation (SHAP),” *International Journal of Computer Networks and Communications*, vol. 13, no. 6, pp. 109–128, 2021, doi: 10.5121/ijcnc.2021.13607.

[35] E. O. Asani, A. Omotosho, P. A. Danquah, J. A. Ayoola, P. O. Ayegba, and O. B. Longe, “A maximum entropy classification scheme for phishing detection using parsimonious features,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 19, no. 5, pp. 1707–1714, Oct. 2021, doi: 10.12928/telkomnika.v19i5.15981.

[36] D. S. Ahmed, A. Hussein, and H. Allah, “Phishing websites detection model based on decision tree algorithm and best feature selection method,” *Turkish Journal of Computer and Mathematics Education*, vol. 13, no. 01, pp. 100–107, 2022, doi: 10.17762/turcomat.v13i1.11964.

BIOGRAPHIES OF THE AUTHORS

Dam Minh Linh was born in Vietnam in 1982. He received an engineering degree in information technology from the Institute of Post and Telecommunications Technology, Vietnam. After that, he received his master’s degree with a major in information systems. He is a lecturer at the Faculty of Information Technology, Institute of Post and Telecommunications Technology in Ho Chi Minh City, Vietnam. His main research areas are IoT for applications, web security, networking security (CORE), machine learning, and computer vision. He can be contacted at email: linhdm@ptit.edu.vn.

Ha Duy Hung received B.S. and M.S. degrees in electronics and telecommunications engineering from Institute of Post and Telecommunication, Vietnam; University of Transport and Communications, Ha Noi, Vietnam in 2007 and 2014. In 2017, he joined the Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Vietnam as a lecturer. In 2021, he received a Ph.D. in communication technology at VSB-Technical University of Ostrava, Czech Republic. His major interests are cooperative communications and physical-layer security. He can be contacted at email: haduyhung@tdtu.edu.vn.

Han Minh Chau was born in Vietnam in 1980, currently the Head of Department Computer Networks, Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam. He received a bachelor’s degree in information technology from Ha Noi University of Science and Technology, a master degree in computer science at the Ho Chi Minh City University of Information Technology, Vietnam National University at Ho Chi Minh City. His main research interests are networking security, machine learning, cloud computing platform, artificial intelligence. He can be contacted at email: hm.chau80@hutech.edu.vn.

Quang Sy Vu was is a lecturer in Faculty of Automotive Engineering, School of Technology, Van Lang University, Ho Chi Minh City, Vietnam. He received the B.E., M.S., and Ph.D. degrees in electrical engineering, all from Peter the Great St. Petersburg State Polytechnical University in 2012, 2014, and 2019 respectively. He can be contacted at email: Sy.vq@vlu.edu.vn.

Thanh-Nam Tran was received the M.Sc. degree in information systems from Military Technical Academy (MTA), in 2014, and the Ph.D. degree in communications technology from the Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava (VSB-TOU), Czech Republic, in 2021. He is currently a Junior Researcher at the Research and Development Team of Prof. Miroslav Voznak, VSB-TOU. He works for the Faculty of Information Technology, Ton Duc Thang University, where he also serves as a Senior Member of the Data Science Laboratory. His research interests include wireless communication networks, data science, and AI. He can be contacted at email: ttnam.it@hotmail.com.