JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 10(2), JUNE, 2022
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

Phishing URLs Classification Using Deep Learning Approach

[1]Hajara Musa, [2]Abdulsalam Ya'u Gital, [3]Abdulkadir Hamidu Alkali, [2]Abubakar Umar, [3]Muhammad Zaharadeen Ahmed
[1]*Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria*
[2]*Department of Mathematical Sciences, Abubakar Tafawa Balewa University Bauchi, Nigeria*
[3]*Department of Computer Engineering, University of Maiduguri, Maiduguri, Nigeria*

ABSTRACT
In recent times, numerous approaches have been proposed for classifying and detecting phishing URLs. While these methods have recorded great successes, most of them are based on content based approaches which has poor generalization ability against new unseen URLS or classical machine learning techniques requiring domain knowledge experts and substantial future engineering. To overcome these limitations, recently, Deep Learning (DL) approaches which offer the ability to automatically capture the semantic or sequential patterns in URLs have been proposed by several studies. This paper reviews and compares the state-of-the-art DL approaches for phishing URLs classification focusing on Recurrent Neural Networks (RNN) and its variants. It has been observed from the experimental results on 60,000 URLs dataset that Gated Recurrent Unit (GRU) model outperforms the basic RNN, and its other variants (LSTM, BiLSTM and, BiGRU) in terms of accuracy, precision, recall, AUC, training, and testing time. Hence the GRU model can be regarded as the first choice for the phishing URL classification in the future work.

ARTICLE INFO
Article History
Received: March, 2022
Received in revised form: May, 2022
Accepted: June, 2022
Published online: June, 2022

KEYWORDS
Phishing, Cybercrime, Deep learning, Dated Recurrent Unit

INTRODUCTION

One of the most common ways to commit cybercrimes is through phishing or malicious URLs. They are employed to host unwanted content and to victimize innocent users, making them victims of several types of internet fraud such as theft of money and identity theft. This has led to financial losses totaling billions of dollars every year and reputation damages among others [1]. Thus, it has become necessary to come up with robust techniques for detecting phishing URLs in a timely manner.

Traditionally, phishing or malicious URLs are detected through blacklisting techniques, which entails compiling a list of known phishing or malicious URLs using crowdsourcing solutions like phishtank [2]. Although these methods just need to perform a quick database search to determine if a given URL is harmful or not and are predicted to have very low false positive rates, their main limitation is that they can only identify malicious URLs that have already been blacklisted. This is a serious concern as new URLs are almost generated on daily basis.

To overcome with these challenges, Machine Learning (ML) techniques which offer the ability to generalize well against new unseen URLs have been employed by several studies [3-7]. While the ML approaches have solved the problem associated with the blacklisting methods, their main limitation is the need for substantial manual features engineering, since they cannot directly learn representations from the URL's sequences of characters [8]. Most of the proposed ML approaches require expert knowledge to determine the best feature set for malicious URL detection and they are unable to effectively capture the sequential patterns in URLs.

To address the above issues, Deep Learning (DL) methods which offer the ability to automatically capture the semantic or sequential patterns in URLs have been proposed by several studies [9-11]. DL is a set of ML algorithms that have the ability to detect

**Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.*

the URL's patterns globally by learning the complex hierarchical feature representations and hidden sequential relationships from large-scale data. These characteristics contribute significantly to the success of DL algorithms in a variety of artificial intelligence (AI) tasks, including speech recognition, image processing, natural language processing, and many more [12].

The Recurrent neural network (RNN) is a type of DL algorithm generally applied for time-series data modeling [13]. The RNN and its variant networks have shown remarkable performance in long-standing AI tasks such as machine translation, language modeling, and speech recognition [12]. However, despite the popularity of RNN in sequence data modeling, to the best of the authors' knowledge, there are few studies on phishing URL classification using the RNN (or its variants) and the utility of various RNN models. Therefore, this paper fills that gap and verifies the performance of different RNN models for malicious URLs classification.

The proposed approaches receive the URL (sequence of characters) as an input and applies the RNNs (RNN, LSTM, GRU, BiLSTM, and BiGRU) at the character level in the URL. The models first identifies the individual characters in the training corpus and then represents each character as a fixed-length vector using one-hot encoding. Using this, the URL sequence characters are converted into an embedding representation, where the models can be applied. Fully connected layer with a sigmoid activation is used to generate the final output of each model. Experimental results showed that all the models obtained great statistical results. However, the GRU model has shown superior performance over the other models in terms of accuracy, precision, recall, AUC, training, and testing time.

The rest of the paper is structured as follows. Section 2 provides a brief introduction of the methodologies including the standard RNN, the LSTM, the GRU, the BiLSTM, and the BiGRU. Section 3 introduces the dataset, evaluation criteria, and experimental design. Section 4 shows the results with relevant discussions. Section 5 draws conclusions.

METHODOLOGIES

RNN

The Recurrent neural network (RNN) is an extension of the traditional feed-forward networks that allow the processing of sequential and time series data. RNN like the traditional feed-forward network consists of three components, input layer, hidden layer, and output layer. However, RNN has an additional internal feedback loop that results in a cyclic graph. These loops are the short-term memory to store and retrieve past information over time scales and thereby making it suitable for the task with temporal dependence [14]. Moreover, while the feed-forward neural networks can only map one input to one output, RNN has the ability of mapping input to output nodes that are, one to many, many to many or many to one [15]. Figure 1 depicts the RNN's fundamental structure. This structure enables the RNN to more accurately describe the temporal dynamic behavior of a time series. That is, an RNN may use the previously learned information to predict the pattern at the current step, which is beneficial in investigating the features of the current time series.

As shown in Figure. 1, xt denotes inputs at time t, St denotes outputs of the hidden layer at time t (named memory at time t), yt denotes outputs of out layer at time t, and [W, U, V] are shared parameters (W denotes the weight of inputs, U denotes the weight of inputs at the current state, and V denotes the weight of outputs). The computational flow of RNN can be formulated mathematically as follows,

$$S_t = f(Ux_t + WS_t - 1 + b_h) \quad (1)$$

$$y_t = f(VS_t + b_o) \quad (2)$$

Where f(.) is an activation function, b_h and b_o are the bias vectors of hidden and out layers respectively.

According to the literature [16], in practice, the RNN cannot maintain a good memory if the time interval is lengthy and there is also a vanishing gradient problem (back-propagation through time approach is usually used for training RNN). As a result, numerous

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

RNN versions are introduced to deal with this problem.

PROCESS FLOW

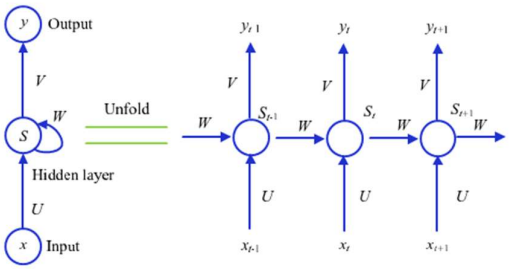

Figure 1: Basic Structure of RNN [14]

LSTM

Long short-term memory (LSTM) is an improvement to the RNN created to solve the vanishing gradients problem experienced when training the conventional RNNs [17]. The architecture of LSTM is composed of memory building blocks that can read, write, and erase data. LSTM uses a gating mechanism to regulate the learning process. *These gates function as memory cells to access and store information over a long period.* The gates and cell state are the LSTM network's basic principles. The cell state is considered the network's memory and serves as the channel for the transmission of relevant information [18]. These gates (i.e., forget, input, and output gates) control the information flow and decide what knowledge should be kept or removed (forgotten), as can be seen in Figure 2. Equations (3)–(8) illustrate the modeling procedures for computing the output of LSTM at step t.

$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ (3)

$f_t = \sigma(U^f x_t + W^f s_{t-1} + b_f)$ (4)

$o_t = \sigma(U^o x_t + W^o s_{t-1} + b_o)$ (5)

$g = \tanh(U^g x_t + W^g s_{t-1} + b_g$ (6)

$c_t = c_{t-1} \cdot f + g \cdot i$ (7)

$s_t = \tanh(c_t) \cdot o$ (8)

Where, W, and b are the trainable weights and biases, respectively, and i, f, and o represent the input gate, forget gate, and output gate respectively. These three gates all share the same shape but with different parameters U and W, which need to be learned during the training process. The hidden state g cannot be used directly. It must pass through the input gate and then be used to calculate the internal storage c_t. While c_t is not only affected by the hidden state but also by c_{t-1}, that is, controlled by the forget gate. Based on c_t, a layer of the *tanh* function is applied to the output information s_t, which is constrained by the output door [19]. The presences of the gates allow the LSTM to satisfy the long-term dependencies in the sequence, and by learning of the gate parameters, the network can determine the appropriate internal storage behaviour.

*Corresponding author: Hajara Musa. mhajara86@gsu.edu.ng Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

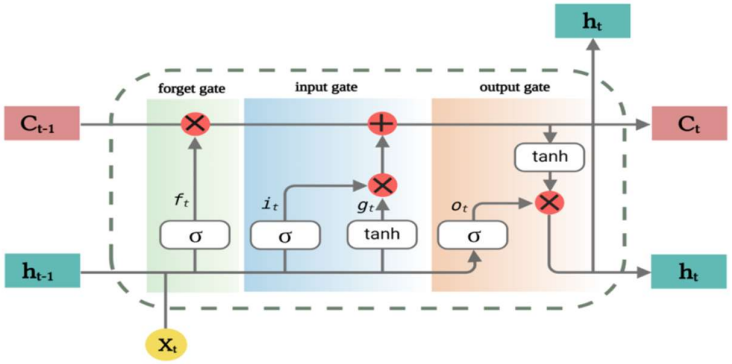

Figure 2: Lstm memory cell [18]

GRU

The Gated recurrent unit (GRU) is another modification of RNN and an alternative to LSTM networks that works in a similar fashion to LSTM. *Like the LSTM, it is capable of effectively retaining long-term dependencies in sequential data. However, compared to the LSTM, GRU has a simpler structure. GRU structure consists of two gates, the Reset Gate and the Update Gate. It merges the Memory and output state [20]. The reset gate (r) controls the information that flows out or is discarded. The forget gate and input gate are combined to form an update gate (z) for regulating the information to be retained from previous memory and also the new memory to be added. By reducing the parameters GRU computation is a little more efficient, simpler, and faster.* A basic GRU structure is shown in Figure. 3. Equations (9) to (12) illustrate the modeling procedure to compute the output of GRU at step t.

$$r_t = \sigma\,(w_r\,.[h_t - 1, x_t] + b_r) \quad (9)$$

$$z_t = \sigma(w_z\,.[h_t - 1, x_t] + b_z) \quad (10)$$

$$g_t = tanh\big(w\,.[r_t \times h_t - 1, x_t] + b_g\big) \quad (11)$$

$$h_t = (1 - z_t) \times h_t - 1 + z_t \times g_t \quad (12)$$

where W and b are the trainable weights and biases respectively, and r and z denote the reset and update gates respectively. g denotes the hidden state which is used to control the removal and addition of new information decided by the update gate and h_t is the output of the GRU at step t.

In general, the LSTM and the GRU both preserve important features using a variety of gating functions, ensuring that they won't be lost when long-term propagation takes place. They can therefore resolve the RNN's vanishing gradient problem through their gate controlling [21]. Additionally, the LSTM has one more gate function than the GRU, making the GRU the simplest RNN structure in this comparative analysis.

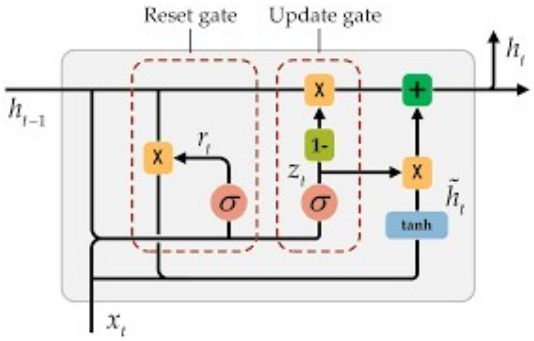

Figure 3: GRU memory cell [24]

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 10(2), JUNE, 2022
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

BILSTM

The bidirectional LSTM (BidLSTM), enhances the conventional LSTM to improve its performance by increasing the amount of information available to the network. The BidLSTM model employs two hidden LSTM layers, one taking the input in a forward direction, and the other in a backward direction to preserve both the future and past information [22]. The idea of a BidLSTM model is quite simple. It involves duplicating the primary recursive layer of the network allowing the network to propagate the input in both forward and backward directions. The input for the primary layer is the actual data during training, whereas the input for the duplicate layer is a reverse copy of the data. This technique effectively increases the amount of information available to the network, thus, improving its performance [23]. Figure 4 indicates the structure of a BidLSTM model. A wrapper for the bidirectional layers required for implementing BidLSTMs networks is provided by the Keras library in Python. Users are given the option to select the merging mode, which controls how the outputs from both directions—that is, forward and backward—are combined before being fed to the subsequent layer. The modeling processes of the forward LSTM layer, the backward LSTM layer, and the output of a BidLSTM network are illustrated briefly as follows (Equations (13) to (15))

$$h_i^1 = f(U^1 \cdot x_i + W^1 \cdot h_{i-1}) \quad (13)$$

$$h_i^2 = f(U^2 \cdot x_i + W^2 \cdot h_{i-1}) \quad (14)$$

$$y_i = softmax(V \cdot [h_i^1; h_i^2]) \quad (15)$$

Where Equation (13) refers to the forward paths and Equation (14) refers to the backward path. y_t is the output of the Bidirectional LSTM obtained by fusing the results from both directional paths. W^1, U^1 and V are weight matrices.

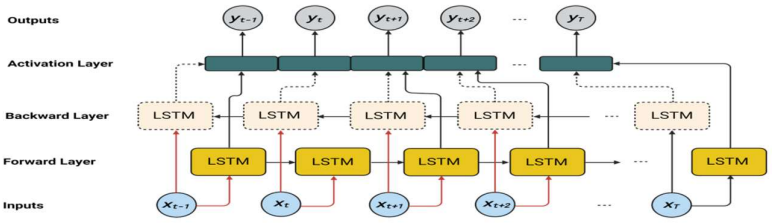

Figure 4: BiLSTM Structure [18]

A **Bidirectional GRU**, or **BiGRU**, is a modified version of the conventional GRU designed to allow the propagation of information in the forward and backward directions. The **BiGRU** like the BiLSTM, consists of two GRUs, one taking the input in a forward direction, and the other in a backward direction to preserve future and past information [24]. The basic idea of the Bi-GRU neural network is that the input sequence is passed through a forward GRU network and a backward GRU network, and then, the outputs of the two are connected in the same output layer [25]. BiGRU effectively increases the amount of information available to the network, improving the context available to the algorithm. Figure 5 indicates the typical structure of a BiGRU. The computational flow of BiGRU can be formulated mathematically using equations 16 to 18 as follows:

$$h_t^{\rightarrow} = GRU_{fw}(x_t, h_{t-1}^{\rightarrow}) \quad (16)$$

$$h_t^{\leftarrow} = GRU_{bw}(x_t, h_{t+1}^{\leftarrow}) \quad (17)$$

$$h_t = h_t^{\rightarrow} \oplus h_t^{\leftarrow} \quad (18)$$

Where is the state of the forward GRU,is the state of the backward GRU, $\oplus$ indicates the operation of concatenating two vectors.

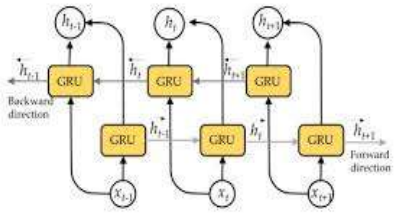

Figure 5: BiGRU Structure [24]

*Corresponding author: Hajara Musa. mhajara86@gsu.edu.ng Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

Modeling phishing URLs Using RNN Networks

To compare different approaches for phishing URL classification, character level embedding has been used in the proposed methodologies. We learn a representation directly from the character sequence of the URL rather than manually extracting the features. Each character sequence exhibits correlations, that is, nearby characters in a URL are likely to be related to each other. These sequential patterns are important because they can be exploited to improve the performance of predictors [26]. RNN-based algorithms, in contrast to other approaches, are able to learn the representation from the URL's character order, eliminating the need for handcrafted feature extraction, which is labor-intensive and requires domain knowledge experts [27]. The proposed networks compose of an embedding layer for URL representation, RNNs layers (RNN, LSTM, GRU, BiLSTM or BiGRU) for optimal feature extraction, and fully connected layers for classification. The embedding layer is usually used as the first layer of the deep neural network structure for a Natural Language Processing (NLP) problem. This layer maps integers of tokens into a fixed multi-dimensional space. A vector returned from the embedding layer stores relations among tokens (single characters for character embeddings, single word for word embeddings) which are fed to the subsequent layer (RNN, LSTM, GRU, BiLSTM or BiGRU) for feature extraction and learning.

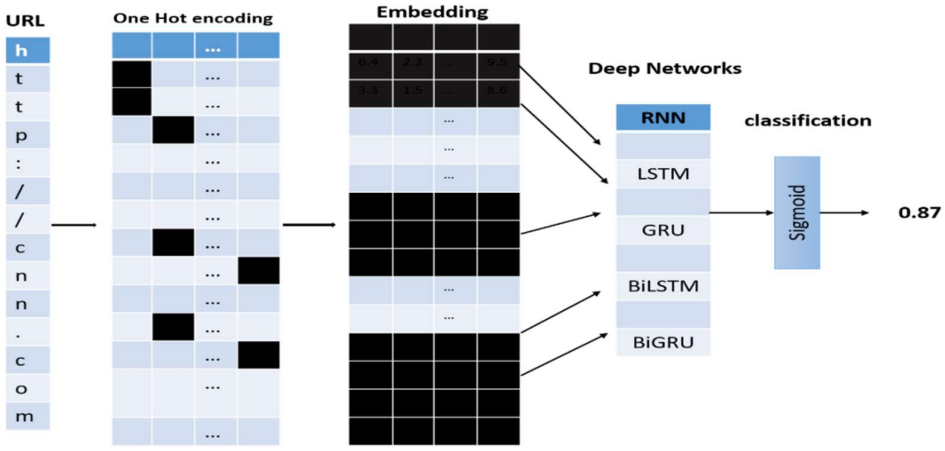

Figure 6: Architecture of the proposed models

Figure. 6 illustrates the architecture of the proposed models. One-hot embedding is used to translate each input, and the 128-dimension vectors are then fed into the RNNs (RNN, LSTM, GRU, BiLSTM, and BiGRU) layers as a 200-step sequence. Finally, the classification is performed using fully connected layer with a sigmoid activation function. The Back-propagation method with the Adam update rule was used for the training of all the proposed networks.

EXPERIMENTAL SETUP

1 Dataset description

To train the proposed models, we constructed a dataset of 30,000 non phishing and 30,000 phishing URLs. In total, sixty thousand URLs were used in the training process. Half of which are legitimates and the other half malicious. The legitimate URLs were crawled from Alexa and DMOZ directory [29-30] while the malicious URLs were crawled from Phishtank [30], a website used as phishing URLs deposit. Five examples of legitimate URLs and five examples of phishing URLs are presented in TABLE I. Note how similar the URLs that are

*Corresponding author: Hajara Musa. mhajara86@gsu.edu.ng Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

malicious and those that are valid might be. This is to be expected given that the attacker's goal is to fool online users by making the phishing site appear as authentic as possible. Moreover, the train test split of the proposed dataset is displayed in Table 2. The crawled URLs are preprocessed and randomly split into training and testing. During preprocessing stage upper-case characters are transformed to lowercase due to distinguishing the upper and lower-case characters.

Table 1: Examples of phishing and non-phishing urls

S/N	URL	Phishing
1	https://www.paypal.com/at/cgi-bin/helpscr?cmd=_security-center-outside	False
2	www.icbc.com	False
3	http://www.alibaba.com/100%2525-lycopene-manufacturers.html	False
3	http://Office365DevAppsModelYam	False
4	https://paypal.com.support-center.mail-accountsupport.center/signin/	True
5	http://office365dbboxs.5gbfree.com/secured-blacklisted/index.php?email=veronique.pedroli@unil.ch	True
6	www.1cbc.com	True

Table 2: Dataset Description

Data	Phishing	Non-phishing	Total
Training	20,000	20,000	40,000
Testing	10,000	10,000	20,000

Experimental design

This paper's main objective is to empirically assess several RNN versions for classifying malicious URLs, not to produce cutting-edge models. Therefore, the setup for our experiments is kept to a minimum, and the comparisons are made fairly. As shown in Table 2, the generated datasets are divided into two separate sets to create the training set and test set, respectively, in order to evaluate the performance of the models. All the models were then trained and evaluated using the same training and testing datasets. The evaluation is done using standard classification evaluation measures, as described below:

$$Accuracy = \frac{TP+}{(TN+TN+TP+FN)} \quad (19)$$

$$Recall = \frac{TP}{(TP+FN)} \quad (20)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (21)$$

$$F-score = \frac{2*(Precision*Recall)}{(Recall+Precis\ )} \quad (22)$$

where TP is the True Positive, which denotes the number of URLs that are phishing and correctly classified as phishing, TN represents True Negative which denotes the number of benign URLs recognized as benign ones, FP represent False Positive, which denotes the number of benign URLs which are wrongly classified as phishing, and FN is the False Negative that denotes the number of phishing URLs identified as benign ones. We define the phishing URLs as positive and the non-phishing URLs as negative. Lastly, we also used the ROC curve to evaluate the AUC Statistics of all the models.

EXPERIMENTAL RESULTS

Table 3: Hyper-parameters

Hyper-parameters	Values
Epochs	20
Learning Rate	0.001
Cell size	128
Batch-Size	256
Dropout Rate	0.2

In this section, we present the experimental results of the proposed models. All the experiments were conducted on a laptop computer with Intel Core (TM) i7-7700HQ CPU @ 2.80GHz x 4, 16GB of DDR3 RAM, and NVIDIA GeForce GTX 1050M 4GB DDR5 GPU. The hyper-parameters used for all models were assigned by hand, and not through hyper-parameter optimization/tuning

*Corresponding author: Hajara Musa. mhajara86@gsu.edu.ng Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

(see Table 3). All models were trained on 40,000 phishing and non-phishing URLs data for 20 epochs. Afterward, the trained models were tested to classify 20,000 phishing and non-phishing URLs. The class distribution of both the training and testing datasets is specified in Table 2.

The performances of the different models in terms of accuracy, precision, recall, and F1-score are summarized in Table 4. As can be seen from Table 4, it can be observed that the GRU and BiLSTM performed best in classifying the URLs as either phishing or non-phishing in comparison to LSTM, BiGRU, and the RNN models. We also evaluated the different models in terms of ROC-AUC (Receiver operating characteristic – Area under the curve) figure 7. Like in the earlier metrics used, GRU and BiLSTM had the best performance (AUC of 0.9998) in comparison to LSTM and BiGRU (AUC of 0.9991) and RNN (AUC of 0.9988). Overall, the performances of all the models in terms of the AUC-ROC curve in classifying malicious URLs are acceptable.

Table 4: Summary of test results

Algorithm	Accuracy	Precision	Recall	F-score
RNN	0.988	0.982	0.989	0.985
LSTM	0.994	0.994	0.992	0.993
GRU	0.995	0.994	0.995	0.995
BiLSTM	0.995	0.994	0.994	0.994
BiGru	0.992	0.989	0.991	0.990

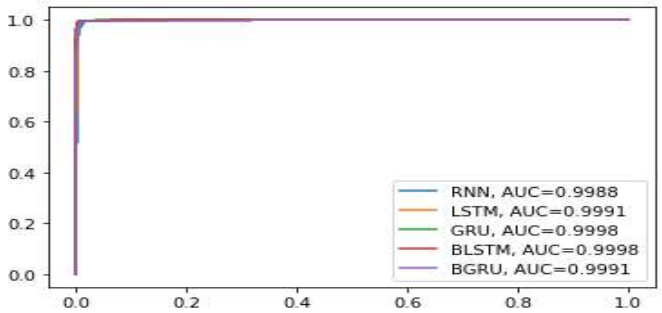

Figure 7: ROC-AUC curves of the proposed models

Furthermore, we studied the training and testing run times of the different models (table 5) in classifying phishing and non-phishing URLs. This was achieved using the time package provided by python. As can be seen from Table 5, the GRU model which finished its training and testing in 152.96 sec and 0.502 sec respectively had the least training and testing time. On the other hand, the RNN model which finished its training and testing in 2403.61 and 2.736 sec had the highest training and testing times. From this result, it can be observed that the training and testing time of the various models is directly related to the complexities of these models. GRU which has the least computational units among the other models has the best training and testing time.

Table 5: Summary of the train and test run times of the algorithms

Algorithm	Training time	Testing time
RNN	2403.605	2.736
LSTM	196.59	0.633
GRU	152.96	0.502
BiLSTM	376.435	1.185
BGru	331.869	0.953

Finally, to visualize the models' performances during training and evaluations, we present the models' learning curves in figure 8 (accuracy vs epoch) and figure 9 (loss vs epoch). As can be seen from both figures, in

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

just few epochs, the validation accuracies and losses of the proposed models converge, increasing the accuracies of each model to over 98% from 10 epochs onward and reducing their losses to below 0.05. BiLSTM, BiGRU, and GRU exhibit constant increase in performance with only slight fluctuations in their accuracies and losses during the validation. On the other hand, RNN and LSTM had fluctuating accuracies and losses during the validation process due to some over-fitting experienced by the two models. However, after 16 epochs, all the models show steady rise in performance untill the 20th epoch. Comparing the training and validation performances of the models, it can be observed that all the models showed constant performances during training and some fluctuations during the validation time, especially with the RNN model. Overall, the training and validation performances of all the models in classifying malicious URLs are acceptable.

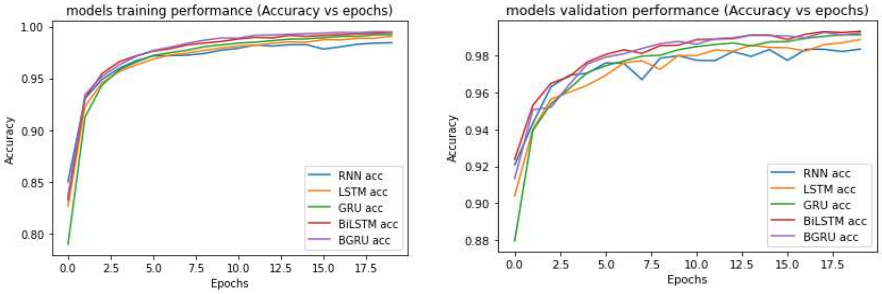

Figure 8: model training and testing performance (accuracy vs epochs)

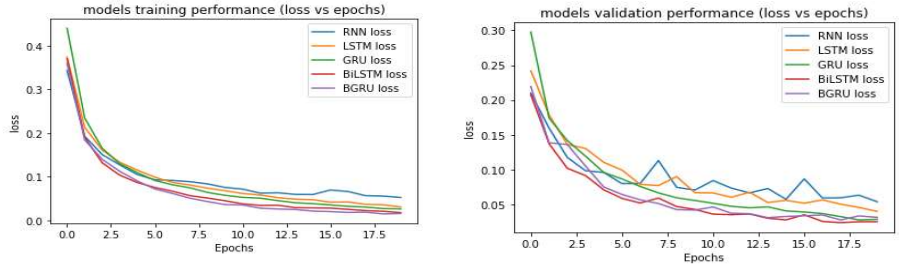

Figure 9: model training and testing performance (loss vs epochs)

DISCUSSIONS AND FUTURE DIRECTIONS

Malicious URL detection is a crucial component of the majority of cyber security applications. Most of the traditional machine learning methods used for malicious URLs detection rely on hand design features, which are labor-intensive to construct and lack some of the attributes necessary to build effective phishing detection systems. To overcome this challenge, recent studies have resorted to the use of deep learning approaches which have the ability to extract the necessary abstract feature representations by acting on raw input data. To leverage on this, in this paper we apply for the task of malicious URLs detection RNN and its variants. The empirical evidence presented suggests that GRU-RNN outperforms RNN and its other variants in the task of phishing and non-phishing URLs classification. However, the results of the other models are also acceptable. Additionally, we have used the various networks with their simple and basic architectures, five layers in each case (three RNNs layers and two dense layers) to avoid high computational cost. To gain more performance, their complex architectures can be explore using advanced hardware in a distributed environment. The characteristics of malicious threats are evolving in nature. At the same time, the URLs are also changing with time. We make a concrete statement such that deep learning mechanisms are most representative to deal with the drifting

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

nature of URLs. In real-time scenarios, getting enough labeled training data is often considered a difficult task. One of the largest available open-source labeled URL's training data is of size 2.4 million [31]. Thus, require a larger study by transforming supervised learning to semi-supervised to unsupervised learning in deep learning mechanisms. This can be considered another significant future direction

CONCLUSION

In this paper, we have explored how well we can discern phishing URLs from legitimate URLs using RNN and its four variants (LSTM, GRU, BiLSTM, and BiGRU) based on character-level embedding. In order to evaluate the approaches, we used a database comprising 30,000 legitimate URLs from the Common Crawl database and 30,000 phishing URLs from Phishtank. The proposed approaches receive the URL (sequence of characters) as an input and applies the RNNs (RNN, LSTM, GRU, BiLSTM, and BiGRU) at the character level in the URL. The models first identifies the individual characters in the training corpus and then represents each character as a fixed-length vector using one-hot encoding. Using this, the URL sequence characters are converted into an embedding representation, where the models can be applied. The embedding layer followed by other deep networks layer extracts features implicitly and those optimal features are fed to other layers of the deeper network for classification in the supervised classification setting. Detailed experiments reveal that all models showed great statistical results. However, the GRU model has shown superior performance over the other models in terms of accuracy, precision, recall, AUC, training, and testing time. This work can be considered preliminary research towards the evaluation of various RNNs for malicious URLs detection and classification. Moreover, adopting deep learning based malicious URL detection in real time should require an extended data sets to avoid the state i.e. an adversary can easily reverse engineer the deep learning mechanisms.

REFERENCES

[1] H. Le, Q. Pham, D. Sahoo, and S. C.H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," *Proceedings of the ACM Symposium on Principles of Distributed Computing*, *Washington, DC*, July 25-27, 2017, 1-13.

[2] PhishTank: An anti-phishing site. Online. Available: https://www.phishtank.com, (Accessed on 5 August, 2022).

[3] A. A. Aminu, A. Abdulkarim, A.Yahaya, and A. M. Turaki, "Detection of phishing websites using Random Forest and XGBoost algorithms," *International Journal of Pure and Applied Sciences,* vol. 2, no. 3 (2019): 1-14.

[4] Z. Erzhou, Z. Chen, J. Cui, and H. Zhong, "MOE/RF: A Novel Phishing Detection Model based on Revised Multi-Objective Evolution Optimization Algorithm and Random Forest." *IEEE Transactions on Network and Service Management,* (2022).

[5] G. Al-Mekhlafi, Z., B. A. Mohammed, M. Al-Sarem, F. Saeed, T. Al-Hadhrami, M. T. Alshammari, A. Alreshidi, and T. S. Alshammari. "Phishing websites detection by using optimized stacking ensemble model." *Computer Systems Science and Engineering* 41, no. 1 (2022): 109-125.

[6] S. Anupam, and A. K. Kar, "Phishing website detection using support vector machines and nature-inspired optimization algorithms," *Telecommunication Systems*,*76*(1), 2021, pp.17-32.

[7] G. Ekta, and D. Gupta, "An efficient approach for phishing detection using machine learning," In *Multimedia Security*, pp. 239-253. Springer, Singapore, 2021.

[8] A. Ali, Q. Jiang, Q. Qu, M. Huang, and J. Niyigena, "An effective

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

phishing detection model based on character level convolutional neural network from URL," *Electronics* 9, no. 9 (2020): 1514.

[9] X. Pingfan, "A Transformer-based Model to Detect Phishing URLs." *arXiv preprint arXiv:2109.02138* (2021).

[10] Z. Erzhou, Q. Yuan, Z. Chen, X. Li, and X. Fang, "CCBLA: a Lightweight Phishing Detection Model Based on CNN, BiLSTM, and Attention Mechanism," *Cognitive Computation* (2022): 1-14.

[11] A. Saad, A. Alotaibi, and O. Alsaleh, "PDGAN: Phishing Detection with Generative Adversarial Networks." *IEEE Access* 10 (2022): 42459-42468.

[12] L. Yann, Y. Bengio, and G. Hinton. "Deep learning." *nature* 521, no. 7553 (2015): 436-444.

[13] J. L. Elman, "Finding structure in time," *Cognitive Science*, 14(2), (1990), 179–211. doi:10.1207/s15516709cog1402_1

[14] B. Su and S. Lu, "Accurate recognition of words in scenes without character segmentation using recurrent neural network," *Pattern Recogn* 2017; 63: 397–405

[15] RNN and its Variants in Handling Sequence Data & Time-Series Data [online], Available:URL, (Accessed on)

[16] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural Comput 2000; 12(10):2451–71.

[17] S. Hochreiter, and J. Schmidhuber,"Long short-term memory," *Neural Computation*, 9(8), (1997), 1735–1780. doi:10.1162/neco.1997.9.8.1735 PMID:9377276

[18] Y. Imrana, Y. Xiang, L. Ali, Z. Abdul-Rauf, Hu, Y.-C. Kadry, S. S. Lim, "χ 2 -BidLSTM: A Feature Driven Intrusion Detection System Based on χ 2 Statistical Model and Bidirectional LSTM," *Sensors* 2022, 22, 2018. https://doi.org/10.3390/s22052018.

[19] W. Youdao, S. Addepalli, and Y. Zhao, "Recurrent neural networks and its variants in remaining useful life prediction," *IFAC-PapersOnLine* 53, no. 3 (2020): 137-142.

[20] C. Kyunghyun, B V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078* (2014)

[21] B. Yun, J. Xie, C. Liu, Y.Tao, B. Zeng, and C. Li, "Regression modeling for enterprise electricity consumption: A comparison of recurrent neural network and its variants," *International Journal of Electrical Power & Energy Systems* 126 (2021): 106612.

[22] M. Schuster, K Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process*. 1997, 45, 2673–2681. [CrossRef]

[23] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada*, 26–31 May 2013; pp. 6645–6649.

[24] L. Xinyu, Y. Wang, X. Wang, H. Xu, C. Li, and X. Xin, "Bi-directional gated recurrent unit neural network based nonlinear equalizer for coherent optical communication system," *Optics Express* 29, no. 4 (2021): 5923-5933.

[25] L. Pengpeng, A. Luo, J. Liu, Y. Wang, J. Zhu, Y. Deng, and J. Zhang, "Bidirectional gated recurrent unit neural network for Chinese address element segmentation," *ISPRS International*

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.

Journal of Geo-Information 9, no. 11 (2020): 635.

[26] R. Tomas, and L. Dovydaitis, "Detection of phishing URLs by using deep learning approach and multiple features combinations," *Baltic journal of modern computing* 8, no. 3 (2020): 471-483.

[27] B. A. Correa, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing URLs using recurrent neural networks," In *2017 APWG symposium on electronic crime research (eCrime)*, pp. 1-8. IEEE, 2017.

[28] http://www.alexa.com/topsites

[29] http://www.dmoz.org/

[30] http://www.phishtank.com/

[31] K. Rieck, T. Krueger and A. Dewald, "Cujo: Efficient detection and prevention of drive-by-download attacks," *In Proceedings of the 26th Annual Computer Security Applications Conference. ACM*, (2010), pp. 31–39.

*Corresponding author: Hajara Musa. ✉ mhajara86@gsu.edu.ng ✉ Department of Mathematical Sciences, Gombe State University, Gombe, Nigeria.