JOURNAL OF ARTIFICIAL INTELLIGENCE AND ROBOTICS
Jul-Sept 2025, VOL. 2, ISSUE 3, pp. 17-24
https://doi.org/10.61577/jaiar.2025.1000013

RESEAPRO JOURNALS

ORIGINAL ARTICLE

OPEN ACCESS

A comparative deep learning approach for phishing URL detection

Abdullahi Musa Yola[1], Hajara Musa[2], Rutarindwa Jean Pierre[1], Mdusbahu Bala Ibrahim[3] and Abdullahi Sani Dauda[3]

[1]Department of Computer Science, Kigali Independent University, Kigali, Rwanda
[2]Department of Computer Sciences, Gombe State University, Gombe, Nigeria
[3]Department of Computer Science, Federal University of Kashere, Gombe, Nigeria

ABSTRACT

Phishing remains a major and serious challenge in this digital era. Recently, deep learning- based approaches have demonstrated superior performance in accurately identifying and classifying phishing URLs. This paper examines and compares modern deep learning methods, with a focus on RNNs and their variants. Moreover, Cybersecurity faces a noteworthy challenge from phishing attacks. Most existing studies rely either on content-based methods, which generalize poorly to new domains and often struggle with previously unseen URLs, or on conventional machine learning methods that require substantial domain expertise and intensive manual feature design. In contrast, deep learning techniques offer a promising alternative for effective phishing URL classification. The Gated Recurrent Unit (GRU) model demonstrated superior performance in phishing URL classification tasks. Its ability to capture sequential patterns effectively makes it a valuable tool for combating phishing attacks in online environments Experimental evaluation conducted on a dataset of 60,000 URLs shows the Gated Recurrent Unit (GRU) model consistently surpasses the standard RNN and other recurrent variants including LSTM, BiLSTM, and BiGRU; in terms of accuracy, precision, recall, AUC, as well as both training and testing efficiency. Further research in this area has strong potential to enhance the effectiveness and efficiency of cybersecurity defenses against phishing attacks. Consequently, the GRU model emerges as a preferred option for future research on phishing URL classification.

KEY WORDS

Deep learning; Gated recurrent unit (GRU); Uniform resource locator classification; Detection; Phishing

ARTICLE HISTORY

Received 11 August 2025;
Revised 02 September 2025;
Accepted 09 September 2025

Introduction

Phishing is a cybercriminal activity where malicious actors fraudulently acquire confidential data, including login credentials, personal account details, and credit card information, by posing as credible organisations across email, messaging services, and other digital communication channels. Traditional email filtering techniques are largely static, making them ineffective against evolving phishing strategies, as attackers constantly adapt their methods to avoid detection [1].

Phishing has developed in diverse and increasingly sophisticated forms, introducing new challenges for accurate detection. Although attackers operate covertly and employ deceptive strategies, security experts and researchers have invested considerable effort in identifying phishing websites. As one of the most common techniques used in cyberattacks, phishing leads to serious consequences, including privacy breaches, identity theft, and financial losses [2]. Phishing strategies have transcended traditional communication mediums, including email and SMS, to encompass a wider array of web-based and mobile platforms. Due to the rapid growth of mobile internet and social networking platforms, phishing activities have extended into new forms, including code-based phishing, spear phishing, and counterfeit mobile applications [3].

Mitigating the threats posed by phishers and other cybercriminals in the online environment necessitates a reliable and automated approach to phishing website detection, as attackers continuously adapt and introduce new methods to carry out their activities. Phishing attackers continuously adapt their techniques to deceive users into disclosing confidential information. Their primary objective is to obtain banking account credentials. Nevertheless, because attackers are highly adaptive and the challenge itself is complex, a fully effective and comprehensive solution has yet to be achieved [4].

Traditionally, phishing URLs have been identified using blocklisting methods, which quickly check a queried database to determine whether a given URL is malicious. Their main shortcoming is that they can only produce very low false positive rates, though they are expected to do so. Identify suspicious websites that are already on the list. These methods are based on a quick database. maliciously check to determine a malicious URL. Although their falsehood is generally very low. Positive rates, the major weakness of such is that it relies on threats identified in the past. This limitation is especially problematic with the ever-growing appearance of malicious URLs on a daily basis.

***Correspondence:** Dr. Abdullahi Musa Yola, Department of Computer Science, Kigali Independent University, Kigali, Rwanda, e-mail: amyola@ulk.ac.rw

In order to overcome these constraints, many researchers have embraced the use of Artificial Intelligence and Machine Learning (ML) methods to make generalizations of already unknown URLs [5-9]. Even though ML-based solutions are used to offset the weakness of conventional blocklisting techniques, they are largely limited because they cannot learn well, and therefore, they depend on large amounts of manual feature engineering. raw character sequence representations [10]. For example, one common handcrafted feature that could be commonly used is the inclusion of the "@" symbol in a URL, a characteristic frequently exploited by scammers to disguise malicious links. Such features cannot be created manually, not to mention that it is not only labor- intensive, but it also demands professional skills to identify the most effective feature set for malicious URL detection. In addition, these characteristics are restricted in their capacity to depict effectively sequential patterns of URLs.

In an effort to curb these challenges, various researchers have proposed Deep Learning (DL) techniques that record semantic or chronological patterns in URLs automatically [11-13]. Deep Learning is a category of machine learning algorithms capable of identifying overarching patterns in URLs by learning intricate hierarchical feature representations and latent sequential representations from default relationships among massive datasets.

A recurrent neural network (RNN) is a form of a Deep Learning algorithm that is typically employed to model. time-series data [14]. Recurrent Neural Networks (RNNs) and their variants have attained significant achievements in artificial intelligence tasks long in existence, including language modeling, machine translation, and speech recognition [13]. However, they are largely used in spite of the fact that they are very common and widely used. Comparatively, there has been little research undertaken in the application of RNN-based sequence modeling. phishing URL classification architectures or studied the relative efficacy of various RNN architectures.

The present study is an attempt to fill this gap by observing and establishing the validity of the. experiments on several variants of RNN in malicious URL classification.

Our model is designed to process URLs as sequential character inputs. This character-level representation is fed into the recurrent layers to facilitate the learning of high-dimensional spatial features character-level. Recurrent Neural Network models, such as RNN, LSTM, GRU, BiLSTM, and BiGRU. The initial phase of the model architecture involves tokenizing the URL strings into individual characters, which are then represented as fixed-length vectors through one-hot encoding. This process transforms the raw categorical data into a numerical format suitable for neural network input. This process converts the character sequences within URLs into an embedding representation that can be effectively utilized by the models. The final architectural component consists of a fully connected (dense) layer utilizing a sigmoid activation function. This configuration maps the high-dimensional features to a probability score, facilitating the binary classification of URLs as either benign or malicious. It has been experimentally tested that all models perform well statistically. However, the architecture based on the GRU performs better than the rest, having better accuracy, recall, AUC, and precision, involving minimal training and experimentation.

Section II presents the methodologies used, including standard GRU, LSTM, RNN, BiLSTM, and BiGRU models. This section gives the background required to the comparative analysis given in the following. sections and underlines that it is essential to choose a proper neural network structure for phishing URL detection. Section 3 Section III explains the data set, evaluation measures, and experimental setup. Section 4 presents the experimental results and provides a detailed discussion of the findings, while Section 5 offers concluding remarks and directions for future work.

Methodologies

Recurrent neural network (RNN)

A Recurrent Neural Network (RNN) is based on the same feed-forward neural network, but it is extended, allowing the modelling of sequential and time-series data. RNN architectures are also similar to feed-forward. However, it incorporates feedback loops that are internal and give a cyclical form of structure that enables information from past time steps to be retained. These feedback loops serve as short-term memory, which facilitates the short-term memory. networks to save and harness past information over time, and this is why RNNs are good at tasks that engage time dependencies [15]. Unlike feed-forward neural networks, which are feed-forward, backward. support multiple types of many-to-many and many-to-one RNNs mapping. input/output structures such as one-to-one, one-to-many, and many-to-many, as well as many-to-one many mappings [16]. The simplest structure of an RNN is presented in Figure 1. This structure allows the network address the problem of effective capture of time dynamics in time-series data through use of previous processes the information previously learned to estimate the patterns at the current time step, thus making forecasting easier. data mining of sequential data.

The structural data flow is illustrated in Figure 1, where xt defines the input vector at a specific interval t. The network's temporal memory is maintained by the hidden state st, while the resulting prediction at that instance is designated as yt. Crucially, the model utilizes a shared parameter framework consisting of three primary weight matrices: W for mapping inputs, U for regulating recurrent hidden state transitions, and V for generating the final layer outputs. The computational procedure of an RNN can be mathematically represented as follows:

$$St = f(Uxt + WSt - 1 + bh) \quad (1)$$

$$yt = f(VSt + bo) \quad (2)$$

f (.) denotes the activation function, and bh and bo represent the bias vectors of the corresponding hidden and output layers, in that order.

Based on the literature, in practice, RNNs frequently have difficulty maintaining long-term dependencies and are prone to vanishing gradients, particularly when trained using backpropagation through time. Consequently, a wide range of RNN variants has been proposed to mitigate these limitations [17].

https://doi.org/10.61577/jaiar.2025.1000013

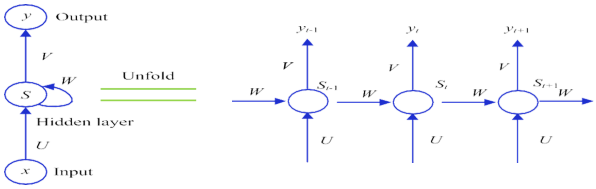

Figure 1. Basic Structure of RNN [15].

Long short-term memory (LSTM)

The LSTM networks are based on the standard RNNs but are extended with the purpose of being more specific. solving the problem of vanishing gradient that is experienced during training [18]. The LSTM architecture is comprised of memory cells that store, update, and discard information as needed. This is done by a gating mechanism that controls the flow of information in the network, which allows storing the appropriate information during the lengthy time periods. The core elements of an LSTM are the cell state and a collection of gates that control its functioning. The cell state functions as the memory component of a network, which offers an avenue of retaining and sending out critical information [19]. The gating framework, which consists of forget, input, and output gates, determines the flow of information since it decides what kind of data to retain or not; it is eliminated, as shown in Figure. 2. The mathematical operations to obtain the LSTM output. At time step t, they are defined in (3) - (8).

$$it = \sigma\,(Wi\cdot[ht-1, xt] + bi) \quad (3)$$

$$ft = \sigma\,(Ufxt + Wfst-1 + bf) \quad (4)$$

$$ot = \sigma\,(Uoxt + Wost-1 + bo) \quad (5)$$

$$g = \tanh\,(Ugxt + Wgst-1 + bg \quad (6)$$

$$ct = ct-1 \cdot f + g \cdot i \quad (7)$$

$$st = \tanh(ct) \cdot o \quad (8)$$

In which Wi and b represent the matrices of trainable weights and bias vectors, respectively, and i, f, and are the input, forget, and output gates. Even though these gates have the same structural form, they are differentiated by a set of parameters acquired during training. The hidden state is not directly available; it is controlled by the input gate and used to compute the internal cell state ct. The current hidden information not only determines the cell state but also the cell's former state, ct–1, under the control of the forget gate. According to the new cell state ct, a tanh activation is then provided, and the resulting st output is gated by the output gate [20]. The gating process makes the LSTM effective at capturing long-term dependencies in sequence data by learning to keep or forget information through appropriate changes in the gate parameters.

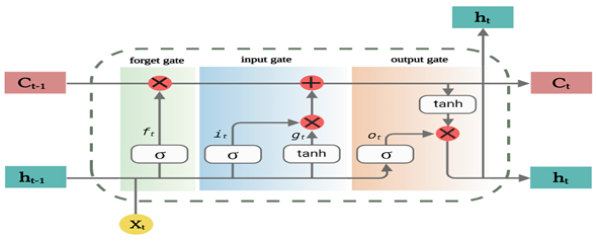

Figure 2. LSTM memory cell [19].

Gated recurrent unit (GRU)

GRU is also a variant of RNN and is an alternative to the LSTM architecture, also based on the same principles. GRUs, just like LSTMs, can model long-term dependencies in sequence data; their design is simpler. The GRU architecture simplifies the gating mechanism by merging the memory and output components into a singular hidden state [21]. This process is governed by two primary gates: the reset gate, which selectively discards irrelevant historical data, and the update gate. The latter functions as a hybrid of 'input' and 'forget' mechanisms, modulating the precise balance between retained information from preceding states and the integration of novel data into the current hidden state. GRU has fewer parameters, enabling it to train faster and more efficiently. Figure. Figure 3 shows the simplest architecture of a GRU. The mathematical expressions are used to determine the GRU output at time step t and are shown in (9)-(12).

$$rt = \sigma\,(wr\,.\,[ht-1, xt] + br) \quad (9)$$

$$zt = \sigma(wz\,.\,[ht-1, xt] + bz) \quad (10)$$

$$gt = tanh(w\,.\,[rt \times ht-1, xt] + bg) \quad (11)$$

$$ht = (1-zt) \times ht-1 + zt \times gt \quad (12)$$

In this context, W and b represent the trainable weight matrices and bias vectors, respectively. The reset gate (r) and update gate (z) control the flow of information; (g) denotes the candidate hidden state. This state governs the integration of new input and the retention of historical data, as determined by the update gate. The final output of the GRU at time step t is represented by ht.

Overall, the LSTM and GRU networks maintain critical features through their gating processes; furthermore, information cannot be lost over time when a sequence is propagated forward. By doing so, they easily resolve the vanishing gradient issue, a common problem in typical RNNs [22]. Moreover, since the LSTM architecture includes an additional gate compared to the GRU, the GRU represents the simplest recurrent neural network structure considered in the present comparative analysis.

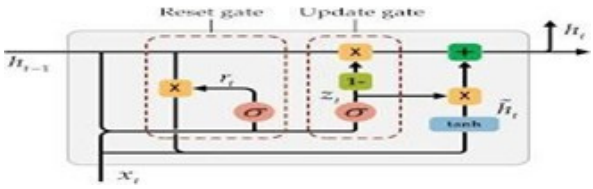

Figure 3. GRU memory cell.

Bidirectional LSTM

BiLSTM builds upon the regular LSTM, wherever the network is able to access more contextual information. It is made up of two LSTMs layers, of which one takes the input sequence in the forward direction and the other in the reverse direction, which allows the model to learn both the past and future dependencies [23]. The BiLSTM architecture is based on a straightforward principle: it duplicates the recurrent layer to allow the network to process input sequences in both forward and backward directions. In this structure, the original layer receives the actual training data, while the replicated layer operates on a reversed version of the same input sequence. This method is known to be efficient at improving the information sent to the network

https://doi.org/10.61577/jaiar.2025.1000013

and thereby enhancing its performance [24]. The architecture of the BiLSTM model is demonstrated in Figure 4. Practically, the Keras library in Python offers a convenient way of implementing bidirectional layers so that it is easy to create BiLSTM networks. The user may define a merge mode to determine how the forward and backward pass results are combined before being sent to the next layer. Basic computational steps involved in the forward LSTM, backward LSTM, and the final BiLSTM output are the following (It is shown in Equations (13) to (15) in brief:

$$h_i^1 = f(U^1 \cdot x_i + W^1 \cdot h_{i-1}) \quad (9)$$

$$h_i^2 = f(U^2 \cdot x_i + W^2 \cdot h_{i-1}) \quad (10)$$

$$y_i = softmax(V \cdot [h_i^1; h_i^2]) \quad (11)$$

In this expression, Equation (13) represents the forward processing path, and Equation (14) represents the reverse processing path. (14) is the reverse way. Yt represents the output of the LSTM/bidirectional LSTM, which is obtained by summing the results of both directional paths. The matrices W1, U1, and V are the weight parameters of the model that are learnable.

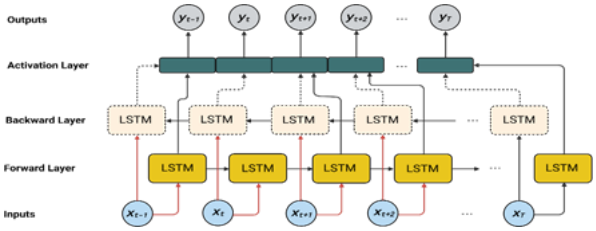

Figure 4. The BiLSTM Structure [19].

The Bidirectional GRU (BiGRU)

A Bidirectional GRU, or BiGRU, is an extension of the traditional GRU, where the direction is two-way information flow. It is made up of two GRU layers that process the input sequence forward and backward. The Backward directions are able to let the model grasp future and past contextual information [25]. The two direction outputs are then added and sent to a common output layer [26]. This bi-directional architecture complements the contextual representation available to the network, resulting in better learning performance. Figure 5 shows the standard design of a BiGRU model. The BiGRU computational process can be mathematically described as follows. The equations (16) - (18) are described as follows:

$$ht\rightarrow = GRUfw(xt, h\rightarrow) \quad (12)$$

$$ht\leftarrow = GRUbw(xt, h\leftarrow) \quad (13)$$

$$ht = ht\rightarrow ht\leftarrow \quad (14)$$

where these represent the states of the forward GRU and the backward GRU, respectively, and ⊕ denotes the concatenation of the two vectors.

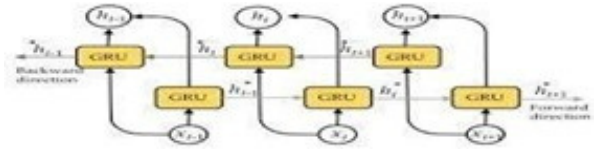

Figure 5. BiGRU Structure [25].

Modelling phishing URLs Using RNN networks

The methods suggested begin with the URL being entered in the form of a series of characters. Initially, each training corpus is characterized and represented as a fixed-length vector with the help of character and one-hot encoding. This method changes the character sequences of the URLs into an embedding representation. These steps can be separated to understand the encoding process clearly before any other modelling is introduced.

After embedding representation comes in the architecture, then uses RNNs at the character level, using various variants of them, e.g., RNN, LSTM, GRU, BiLSTM, BiGRU. After the data passes through the RNN layers, it is sent to a fully connected layer followed by a sigmoid activation function, which serves the purpose of generating the final output. This graded method is such that it makes the readers able to follow the pipeline step by step, lessening the cognitive load and getting to know the complex methodology.

Logistic regression

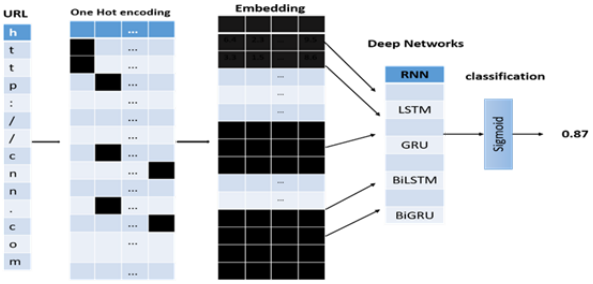

Figure 6. Architecture of the proposed models.

As shown in Figure 6, the design of our models follows a structured pipeline. We begin by transforming the raw input data via one-hot embedding, which produces 128-dimensional vectors. These are then processed as a 200-step sequence by the various recurrent architectures specifically RNN, LSTM, GRU, BiLSTM, and BiGRU. To reach a final classification, the data passes through a fully connected layer governed by a sigmoid activation function. For the learning phase, we used backpropagation with the Adam optimizer to refine all network parameters.

Experimental setup

Dataset description

The experimental dataset consists of 60,000 samples, comprised of 30,000 legitimate and 30,000 phishing URLs. This balanced distribution ensures an equal representation of benign and malicious classes. Legitimate instances were extracted from the Alexa and DMOZ directories [27-29], while malicious samples were retrieved from PhishTank, a reputable repository for verified phishing links. Table 1 presents five examples each of legitimate and phishing URLs. Malicious URLs can closely mimic legitimate ones, which aligns with attackers' goal of misleading users into believing phishing sites are authentic. The distribution of the proposed dataset across the training and test partitions is detailed in Table 2. Following a pre-screening process, the collected URLs were randomly allocated into their respective sets. To ensure consistency during preprocessing, all character data underwent normalization by converting uppercase letters to lowercase [30].

https://doi.org/10.61577/jaiar.2025.1000013

Table 1 . 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
4	http://Office365DevAppsModelYam	False
5	https://paypal.com.support-center.mail-accountsupport.center/signin/	True
6	http://office365dbboxs.5gbfree.com/secured- blacklisted/index.php?email=veronique.pedroli@unil.ch	True
7	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 study primarily aims to evaluate several variations of RNNs on malicious URL classification as opposed to the suggestion of the state-of-the-art models analytically. Consequently, the experimental setup is maintained as minimally as possible in order to make fair comparisons. As detailed in Table 2, the data were partitioned into training and testing sets to evaluate model performance. These respective subsets were subsequently utilized to develop and validate all experimental models. It should be noted that very little hyperparameter tuning was actually done among the models, so that the assessment process is not focused on model architecture but fine- tuning accreditations. The reason this decision was made is the reduction of the differences in optimization because different hyperparameters might discriminate against the fairness of a comparison between models. Through the application of a set of hyperparameters, which is held constant, we are able to establish a fair comparison scheme that protects against any prior bias of the optimization equality of the models. Assessment is done based on standard classification performance metrics are defined as follows:

$$(15) \quad Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)}$$

$$(16) \quad Recall = \frac{TP}{(TP+FN)}$$

$$(17) \quad Precision = \frac{TP}{(TP+FP)}$$

$$(18) \quad F-score = \frac{2*(Precision*Recall)}{(Recall+Precision)}$$

Experimental design

Table 3 . Hyperparameters.

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

The experiments were carried out on a laptop equipped with an Intel Core i7-7700HQ processor running at 2.80 GHz (4 cores), 16 GB of DDR3 RAM, and an NVIDIA GeForce GTX 1050M GPU with 4 GB of GDDR5 memory. The hyperparameters of all models were not optimized automatically by using automated tuning methods and instead they were chosen manually as it is presented in Table 3. All the models were trained over 20 epochs on a 40,000 URL dataset consisting of phishing and non-phishing URLs. Evaluation was then done on another test set of 20,000 URLs using the trained models. Table 2 gives the class representation of the training and testing sets.

Table 4 provides a summary of the models' comparative performance based on accuracy, precision, recall, and F1-score. The GRU and BiLSTM models have better classification results in the task of differentiating between phishing and legitimate URLs, as demonstrated in the table, and performed better than the LSTM, BiGRU, and the standard RNN models. Moreover, the models were evaluated based on the ROC-AUC measure and its results were depicted in Figure 7. In line with the previous assessment measures, the values of AUC align the highest with GRU and BiLSTM (0.9998), then LSTM and BiGRU (0.9991), and finally with the RNN model (0.9988). In general, the results of AUC-ROC prove that all proposed models show good results in identifying malicious URLs.

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

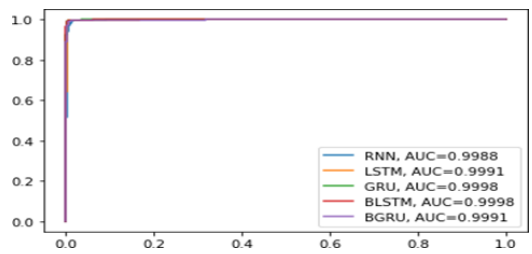

Figure 7. ROC-AUC curves of the proposed models.

https://doi.org/10.61577/jaiar.2025.1000013

The training and testing runtimes of the various models were also analyzed, as indicated in Table 5, to classify phishing and non-phishing URLs. The Python time package was used to get the runtime measurements. The GRU model was the fastest model, as indicated in Table 5, since the model took 152.96seconds to trainand 0.502 secondsto test. Conversely, the conventionalRNN model had the highest run times with a requirement of. Training and test times were 2403.61 and 2.736 seconds, respectively. The GRU model has the best runtime performance among the tested models, thanks to its relatively simple architecture and the number of computational units.

Table 5 . The 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
BiGru	331.869	0.953

This demonstrates the model's performance during training and evaluation; the learning curves are shown in Figure 8 (accuracy versus epoch) and Figure 9 (loss versus epoch). As observed in both figures, the loss and validation accuracy of the proposed models converge within a few epochs. Since the tenth epoch,the validation accuracyof all models has been above 98%, withthe loss falling below 0.05. The performance can be observed to be stable in the BiLSTM, BiGRU, and GRU models, with insignificant alterations in accuracy and loss of validation. On the other hand, there are observable variations in the RNN and LSTM models in the validation process, which can be explained by overfitting. However,from the sixteenthepoch onwards, all the models have a linear enhancement of performance untilthe twentieth epoch.Overall, the trainingand validation results demonstrate that all proposed models achievestrong performance in classifying malicious URLs.

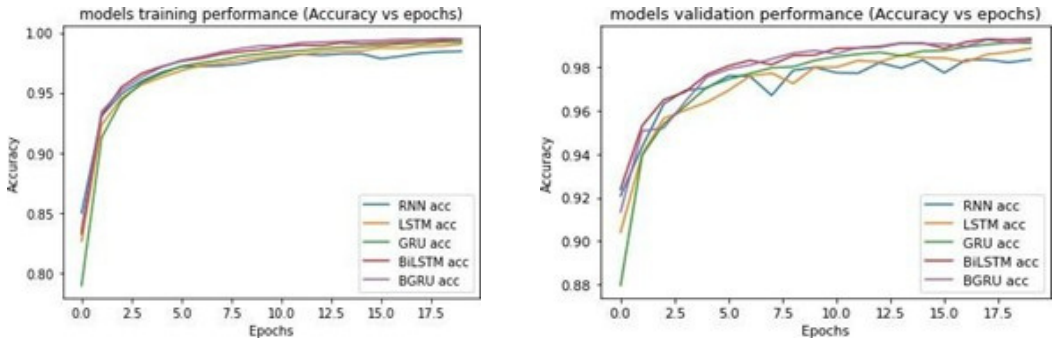

Figure 8. Illustrates model training and testing performance (accuracy vs epochs).

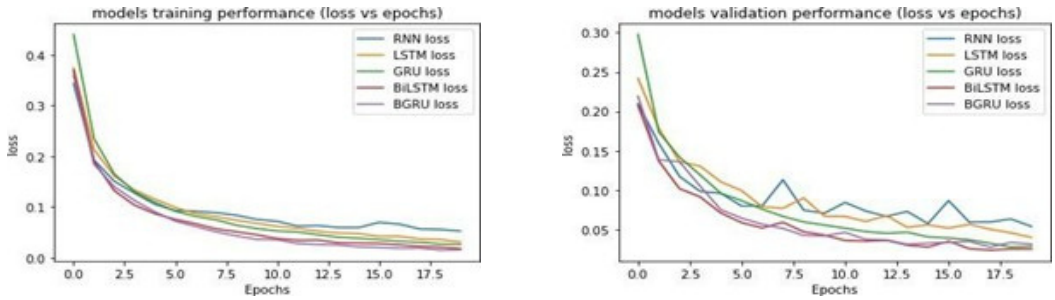

Figure 9. The model training and testing performance (loss vs epochs).

Discussions and future directions

Critical URL detection is an important part of the contemporary cybersecurity mechanism. Conventional machine learning methods to perform this task are usually based on manually designed features, which require significant time to design and are not as complicated as successful phishing detection demands [31]. In order to mitigate these shortcomings, recent studies have been shifting towards a more recent shift in the development of deep learning methods, which are able to learn abstract feature representations directly out of raw data. This paper will examine the use of RNNs and their derivatives to detect malicious URLs. Depending on the results of the experiment, it can be stated that the GRU-based RNN outperforms the conventional RNN and other models, but the overall

https://doi.org/10.61577/jaiar.2025.1000013

performance of all the considered models can be considered satisfactory [32]. To reduce the computational cost, comparatively uncomplicated network designs were used, and every model had 5 layers, three recurrent layers, and two fully connected layers. Although this design decision is efficient, more may be done to enhance the performance through experimenting with more complicated architectures using sophisticated hardware platforms in a distributed computing setting. Likewise, it should also be noted that threats of a malicious nature keep changing, and the URL patterns keep changing as time progresses. In this respect, deep learning techniques can specifically cope with the concept drift that URL data involves. In practice, obtaining large labeled datasets that are large enough to train is a major problem. Though the dataset of labeled URLs available publicly has about 2.4 million samples, additional studies to go beyond fully supervised learning are necessary [33]. Further research should then be carried out on semi- supervised and unsupervised deep learning models, which are potential avenues that make the malicious URL detection systems more robust and scalable.

Conclusion

The paper steered an experiment to determine the performance of RNNs and their sub-types, LSTM, GRU, BiLSTM, and BiGRU in predicting phishing URLs and legal ones by using character-level embeddings. It was tested on the dataset of 30,000 legitimate URLs that were gathered in Common Crawl and 30,000 phishing URLs that were gathered in Phishtank. Both methods consider a URL as a sequence of characters and perform recurrent architectures on the character-level, where each character in the training corpus is initially recognized and converted into fixed-length vectors by one-hot encoding. The process encodes the URL character sequences into embedding representations, which are then fed into deep network layers that automatically learn useful features and make a supervised classification. As shown in experiments, all four of the evaluated models have a high level of statistical performance, but the GRU-based model has consistently better performance in terms of accuracy, precision, recall, AUC, as well as training and testing efficiency compared to the other models. This study serves as a preliminary investigation into RNN-based approaches for detecting malicious URLs. Furthermore, real-time phishing detection systems leveraging deep learning models require large datasets to effectively mitigate the associated threats, manipulation, and reverse engineering of models. Practically, in terms of cybersecurity, the results indicate that GRU models must be selected to classify phishing URLs in real-time because of their good balance in performance as well as computational efficiency.

Disclosure Statement

No potential conflict of interest was reported by the authors.

References

1. Huang W, Chu DT, Bai LY, Kang W, Zhang HT, Li B, et al. EvoMail: Self-Evolving Cognitive Agents for Adaptive Spam and Phishing Email Defense. arXiv preprint arXiv:2509.21129. 2025. https://doi.org/10.48550/arXiv.2509.21129
2. Alkhalil Z, Hewage C, Nawaf L, Khan I. Phishing attacks: A recent comprehensive study and a new anatomy. Front Comput Sci. 2021;3:563060. https://doi.org/10.48550/arXiv.2509.21129
3. Shahriar H, Klintic T, Clincy V. Mobile phishing attacks and mitigation techniques. J Inf Secur. 2015;6(3):206. http://dx.doi.org/10.4236/jis.2015.63021
4. Yadollahi MM, Shoeleh F, Serkani E, Madani A, Gharaee H. An adaptive machine learning based approach for phishing detection using hybrid features. In2019 5th International Conference on Web Research (ICWR) 2019:281-286. https://doi.org/10.1109/ICWR.2019.8765265
5. Shahriar H, Klintic T, Clincy V. Mobile phishing attacks and mitigation techniques. J Inf Secur. 2015;6(3):206. http://dx.doi.org/10.4236/jis.2015.63021
6. Le H, Pham Q, Sahoo D, Hoi SC. URLNet: Learning a URL representation with deep learning for malicious URL detection. arXiv preprint arXiv:1802.03162. 2018. https://doi.org/10.48550/arXiv.1802.03162
7. PhishTank: An anti-phishing site. 2022. Available at: https://www.phishtank.com
8. Dharani M, Badkul S, Gharat K, Vidhate A, Bhosale D. Detection of phishing websites using ensemble machine learning approach. InITM web of conferences 2021;40:03012. https://doi.org/10.1051/itmconf/20214003012
9. Zhu E, Chen Z, Cui J, Zhong H. MOE/RF: a novel phishing detection model based on revised multiobjective evolution optimization algorithm and random forest. IEEE Transactions on Network and Service Management. 2022;19(4):4461-4478. https://doi.org/10.1109/TNSM.2022.3162885
10. Ghaleb Al-Mekhlafi Z, Abdulkarem Mohammed B, Al-Sarem M, Saeed F, Al-Hadhrami T, Alshammari MT, Alreshidi A, et al. Phishing websites detection by using optimized stacking ensemble model. Computer Systems Science and Engineering. 2022;41(1):109-125. https://doi.org/10.32604/csse.2022.020414
11. Anupam S, Kar AK. Phishing website detection using support vector machines and nature-inspired optimization algorithms. Telecommunication Systems. 2021;76(1):17-32. https://doi.org/10.1007/s11235-020-00739-w
12. Gandotra E, Gupta D. An efficient approach for phishing detection using machine learning. InMultimedia security: algorithm development, analysis and applications 2021:239-253. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-8711-5_12
13. Aljofey A, Jiang Q, Qu Q, Huang M, Niyigena JP. An effective phishing detection model based on character level convolutional neural network from URL. Electronics. 2020;9(9):1514.
14. Zhu E, Yuan Q, Chen Z, Li X, Fang X. CCBLA: a lightweight phishing detection model based on CNN, BiLSTM, and attention mechanism. Cogn Comput. 2023;15(4):1320-1333. https://doi.org/10.1007/s12559-022-10024-4
15. Al-Ahmadi S, Alotaibi A, Alsaleh O. PDGAN: Phishing detection with generative adversarial networks. Ieee Access. 2022;10:42459-42468. https://doi.org/10.1109/ACCESS.2022.3168235
16. LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436-444. https://doi.org/10.1038/nature14539
17. Elman JL. Finding structure in time. Cogn Sci. 1990;14(2):179-211. https://doi.org/10.1207 s15516709cog1402_1
18. Su B, Lu S. Accurate recognition of words in scenes without character segmentation using recurrent neural network. Pattern Recognit. 2017;63:397-405. https://doi.org/10.1016/j.patcog.2016.10.016

https://doi.org/10.61577/jaiar.2025.1000013

19. Wang J, Li X, Li J, Sun Q, Wang H. NGCU: A new RNN model for time-series data prediction. Big Data Research. 2022;27:100296. https://doi.org/10.1016/j.bdr.2021.100296
20. Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural Comput. 2000;12(10):2451-2471. https://doi.org/10.1162/089976600300015015
21. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
22. Imrana Y, Xiang Y, Ali L, Abdul-Rauf Z, Hu YC, Kadry S et al. χ 2-bidlstm: a feature driven intrusion detection system based on χ 2 statistical model and bidirectional lstm. Sensors. 2022;22(5):2018. https://doi.org/10.3390/s22052018
23. Wang Y, Addepalli S, Zhao Y. Recurrent neural networks and its variants in remaining useful life prediction. IFAC-PapersOnLine. 2020;53(3):137-142. https://doi.org/10.1016/j.ifacol.2020.11.022
24. Cho K, Van Merriënboer B, Gulçehre Ç, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014:1724-1734. Available at: https://aclanthology.org/D14-1179.pdf
25. Bai Y, Xie J, Liu C, Tao Y, Zeng B, Li C. Regression modeling for enterprise electricity consumption: A comparison of recurrent neural network and its variants. Int. J. Electr. Power Energy Syst. 2021;126:106612. https://doi.org/10.1016/j.ijepes.2020.106612
26. Schuster M, Paliwal KK. Bidirectional recurrent neural networks. IEEE transactions on Signal Processing. 1997;45(11):2673-2681. https://doi.org/10.1109/78.650093
27. Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent neural networks. In2013 IEEE international conference on acoustics, speech and signal processing 2013:6645-6649. https://doi.org/10.1109/ICASSP.2013.6638947
28. Liu X, Wang Y, Wang X, Xu H, Li C, Xin X. Bi-directional gated recurrent unit neural network based nonlinear equalizer for coherent optical communication system. Optics Express. 2021;29(4):5923-5933. https://doi.org/10.1364/OE.416672
29. Li P, Luo A, Liu J, Wang Y, Zhu J, Deng Y, et al. Bidirectional gated recurrent unit neural network for Chinese address element segmentation. Int. J. Geo-Inf. 2020;9(11):635. https://doi.org/10.3390/ijgi9110635
30. Rasymas T, Dovydaitis L. Detection of phishing URLs by using deep learning approach and multiple features combinations. Balt J Mod Comput. 2020;8(3):471-483. https://doi.org/10.22364/bjmc.2020.8.3.06
31. Bahnsen AC, Bohorquez EC, Villegas S, Vargas J, González FA. Classifying phishing URLs using recurrent neural networks. In2017 APWG symposium on electronic crime research (eCrime). 2017:1-8. IEEE. https://doi.org/10.1109/ECRIME.2017.7945048
32. Rieck K, Krueger T, Dewald A. Cujo: efficient detection and prevention of drive-by-download attacks. InProceedings of the 26th Annual Computer Security Applications Conference 2010:31-39. https://doi.org/10.1145/1920261.1920267
33. Ahammad SH, Kale SD, Upadhye GD, Pande SD, Babu EV, Dhumane AV, Bahadur MD. Phishing URL detection using machine learning methods. Adv Eng Softw. 2022;173:103288. https://doi.org/10.1109/AIIoT61789.2024.10579005

https://doi.org/10.61577/jaiar.2025.1000013