Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057138

ARTICLE

A Hybrid CNN-Brown-Bear Optimization Framework for Enhanced Detection of URL Phishing Attacks

Brij B. Gupta[1,*], Akshat Gaurav[2], Razaz Waheeb Attar[3], Varsha Arya[4], Shavi Bansal[5], Ahmed Alhomoud[6] and Kwok Tai Chui[7]

[1]Department of Computer Science and Information Engineering, Asia University, Taichung, 413, Taiwan

[2]Computer Engineering, Ronin Institute, Montclair, NJ 07043, USA

[3]Management Department, College of Business Administration, Princess Nourah bint Abdulrahman University, Riyadh, 11671, Saudi Arabia

[4]Department of Business Administration, Asia University, Taichung, 413, Taiwan

[5]Department of Research and Innovation, Insights2Techinfo, Jaipur, 302001, India

[6]Department of Computer Science, College of Science, Northern Border University, Arar, 91431, Saudi Arabia

[7]Department of Electronic Engineering and Computer Science, Hong Kong Metropolitan University (HKMU), Hong Kong, China

*Corresponding Author: Brij B. Gupta. Email: bbgupta@asia.edu.tw

Received: 09 August 2024 Accepted: 26 November 2024 Published: 19 December 2024

ABSTRACT

Phishing attacks are more than two-decade-old attacks that attackers use to steal passwords related to financial services. After the first reported incident in 1995, its impact keeps on increasing. Also, during COVID-19, due to the increase in digitization, there is an exponential increase in the number of victims of phishing attacks. Many deep learning and machine learning techniques are available to detect phishing attacks. However, most of the techniques did not use efficient optimization techniques. In this context, our proposed model used random forest-based techniques to select the best features, and then the Brown-Bear optimization algorithm (BBOA) was used to fine-tune the hyper-parameters of the convolutional neural network (CNN) model. To test our model, we used a dataset from Kaggle comprising 11,000+ websites. In addition to that, the dataset also consists of the 30 features that are extracted from the website uniform resource locator (URL). The target variable has two classes: "Safe" and "Phishing." Due to the use of BBOA, our proposed model detects malicious URLs with an accuracy of 93% and a precision of 92%. In addition, comparing our model with standard techniques, such as GRU (Gated Recurrent Unit), LSTM (Long Short-Term Memory), RNN (Recurrent Neural Network), ANN (Artificial Neural Network), SVM (Support Vector Machine), and LR (Logistic Regression), presents the effectiveness of our proposed model. Also, the comparison with past literature showcases the contribution and novelty of our proposed model.

KEYWORDS

Phishing attack; CNN; brown-bear optimization

1 Introduction

The term "Phishing" is derived from the term "Fishing" because phishing attacks use "bait" (attractive messages) to lure the victim and then "fish" for the personal information they want to steal [1]. The first phishing attack was reported in 1995, in which attacks steal the victim's account details [1]. Since then, phishing attacks have been considered one of the most harmful cyber attacks [2–4]. In this context, Fig. 1 presents the number of phishing websites from 2013 to 2022 [5]. The number of phishing websites has increased during COVID-19 due to increased digitization. Also, Fig. 2 presents the impact of phishing attacks on different sectors [6]. From Fig. 2, it is clear that the phishing attack is a universal problem that needs efficient and accurate solutions.

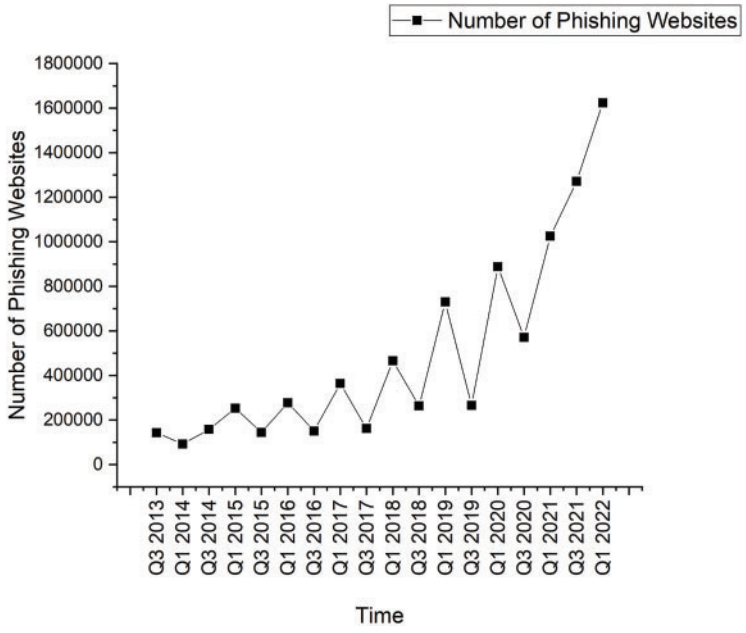

Figure 1: Number of phishing websites detected

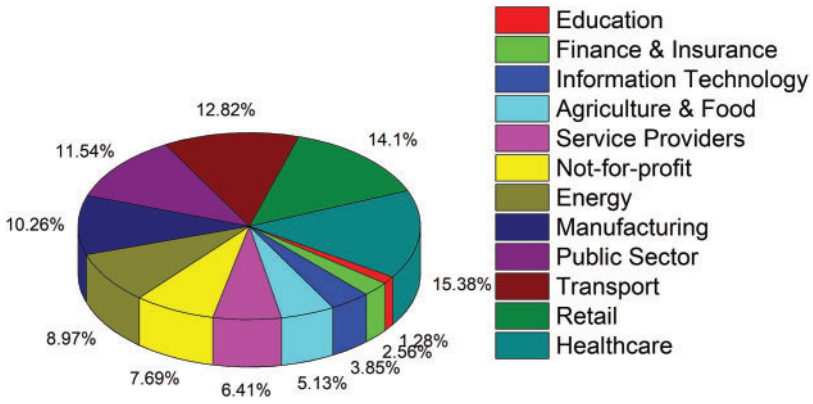

Figure 2: Phishing attack in different sectors

In the phishing attack using the malicious Uniform Resource Locator (URL), an attacker sends the phishing link to the victim, which leads him to the phishing website hosting on the compromised

web server [7–10]. This malicious website is used to steal the valuable assets of the victim [11]. There are two main approaches for the detection of phishing URLs. The first is the blocklisting approach, and the other is the artificial intelligence-based approach [12]. In the first approach, a blocklist is prepared to help users identify phishing URLs. However, this method is not economical because attackers use different URLs to create a website. The second technique used deep learning and machine learning models to identify phishing URLs [13].

1.1 Research Gap

Recently, researchers have been proposing deep learning models for detecting phishing websites with the help of URLs; however, most of the approaches show a degradation in the recall values after the phishing attack is recorded. Hence, the current limitation of the phishing attack detection models is the low recall value [7]. In addition, selecting the most relevant features from the URLs is also an important factor of an efficient phishing attack detection model [14]. Moreover, there is a research gap in understanding how hyper-parameter tuning can improve the performance of the deep learning model [15].

1.2 Contribution

As there are not many deep learning models that use efficient optimization techniques, in this paper, we proposed a hybrid CNN-brown-bear optimization (BBOA) framework for enhanced detection of URL phishing attacks. Our proposed framework uses random forest to extract relevant features, and then BBOA is used to train the CNN model. We only used a single layer of CNN and trained our model for five epochs, which makes our proposed model lightweight and less complex.

We used random forest for feature extraction because it offers robustness against overfitting, handles high-dimensional data well, and can automatically capture non-linear interactions between features, providing a reliable assessment of feature importance. The BBOA is known for its strong global search capability and robustness in diverse problem landscapes, effectively balancing exploration and exploitation to avoid local optima. Due to these advantages, our proposed approach outperforms the other standard deep learning and machine learning models and performs better than current phishing detection models.

1.3 Paper Organization

The rest of the paper is organized as follows:

- Section 2 presents the related work.
- Section 3 gives details about the proposed approach.
- Section 4 presents the simulation results.
- Section 5 concludes the paper.

2 Related Work

Liu et al. [16] propose a novel approach called TransURL for malicious URL detection, utilizing a character-aware Transformer combined with three feature modules-Multi-Layer Encoding, Multi-Scale Feature Learning, and Spatial Pyramid Attention. This design enhances the extraction of character-level information and structural relationships from URLs. The model significantly improves class-imbalanced, multi-classification, cross-dataset testing, and adversarial scenarios. However, the

limitations include potential complexity in training due to the multi-module structure and challenges in generalizing to diverse URL patterns not covered in the training data.

Liu et al. [17] propose PMANet, a pre-trained Language Model-Guided multi-level feature attention network designed to enhance malicious URL detection by addressing limitations such as domain adaptability, character-level information, and local detail extraction. PMANet introduces a novel post-training program using self-supervised learning objectives and a layer-wise attention mechanism, effectively integrating hierarchical representations of URLs. However, the limitations include the potential for high computational costs associated with the multi-layer attention mechanism and the reliance on extensive training data for effective adaptation to the URL domain.

Bu et al. [7] proposed a deep learning-based phishing website detection technique that used genetic algorithms to select optimal features. The focus of the proposed approach is to increase the recall value. Bu et al. [14] proposed a phishing URL detection approach using deep convolutions autoencoder and character-level URL features extraction. However, the author didn't use any feature selection and optimization techniques.

Kamble et al. [2] proposed a SqueezeNet-based phishing website detection technique. The SqueezeNet model is optimized using Hunter Prey Optimization (FDHPO). The proposed model selects optimal features from the web, ocular, and Natural Language Processing (NLP) features. However, using the SqueezeNet model increased the overall complexity of the proposed model. Also, sometimes SqueezeNet leads to over-fitting. Alamosa et al. [15] used a systematic approach to build optimized deep learning to detect phishing websites. The proposed deep learning model is based on LSTM, ANN, and CNN; the model is optimized by grid search and genetic algorithm. However, the proposed approach is complex due to the use of grid search and genetic algorithms. Rani et al. [18] proposed a machine learning model for phishing website detection with the help of website URLs. The author used TreeSHAP and Information Gain to extract the best features. Then, the extracted features are used to predict the phishing URLs using Naive Bayes, Random Forest, and XGBoost techniques. However, using TreeSHAP increased the proposed model's computational cost.

Prabakaran et al. [11] proposed variational autoencoders (VAE) and deep neural networks (DNN) based approach for the detection of URL-based phishing attacks. However, VAEs can struggle with capturing high-frequency details in data, making them less effective for tasks requiring precise feature extraction. Sahingoz et al. [19] proposed a bat-based optimization algorithm for detecting phishing websites. However, the bat algorithm-based optimization method can suffer from premature convergence in complex search spaces and struggle with scalability when applied to significant or high-dimensional problems. Adane et al. [20] proposed a stack-based model for detecting phishing websites. The proposed approach extracts the features from Uni-variate feature selection (UFS). However, UFS may miss the interaction between the features because it only evaluates each feature independently. Sahingoz et al. [21] proposed a real-time-based phishing website attack detection model. The proposed model used seven machine learning models of natural language processing (NLP) for feature extraction. However, NLP-based feature extraction can struggle with understanding context and nuances in language; hence, it is inefficient in extracting features. Saha et al. [22] presents a data-driven method for detecting phishing webpages using a deep learning approach. The proposed method used the primary method for feature extraction but did not use hyperparameter tuning.

Jain et al. [23] proposed the detection of phishing websites by analyzing the URLs in the HTML source code. The extracted URLs are differentiated into 12 different features, and then these features are used to train the machine learning model. However, the proposed approach did not use the optimization techniques. Yang et al. [24] proposed a multidimensional feature detection approach

based on deep learning. The proposed approach is two-phased: in the first phase, the sequence-based features of the URL are extracted; in the second phase, URL statistical features, webpage text features, and webpage code features are extracted for the classification. The author used a threshold to select the second phase. Mourtaji et al. [25] proposed a phishing URLs detection model based on six different feature extraction methods, including the blocked method, lexical and host method, content method, identity method, identity similarity method, visual similarity method, and behavioral method. Then, the author used machine learning and deep learning-based models to identify malicious URLs.

In summary, there are many deep learning and machine learning models for detecting phishing URLs; however, they do not use optimization methods efficiently. Some did not use feature and hyper-parameter extract techniques, and others did not use efficient optimization techniques. This is a research gap that is covered in this paper.

3 Proposed Approach

This section presents the details of the proposed framework. Fig. 3 presents the details of the model. The framework leverages feature selection, a convolutional neural network (CNN), and the Brown-Bear Optimization Algorithm (BBOA) for efficient and accurate detection.

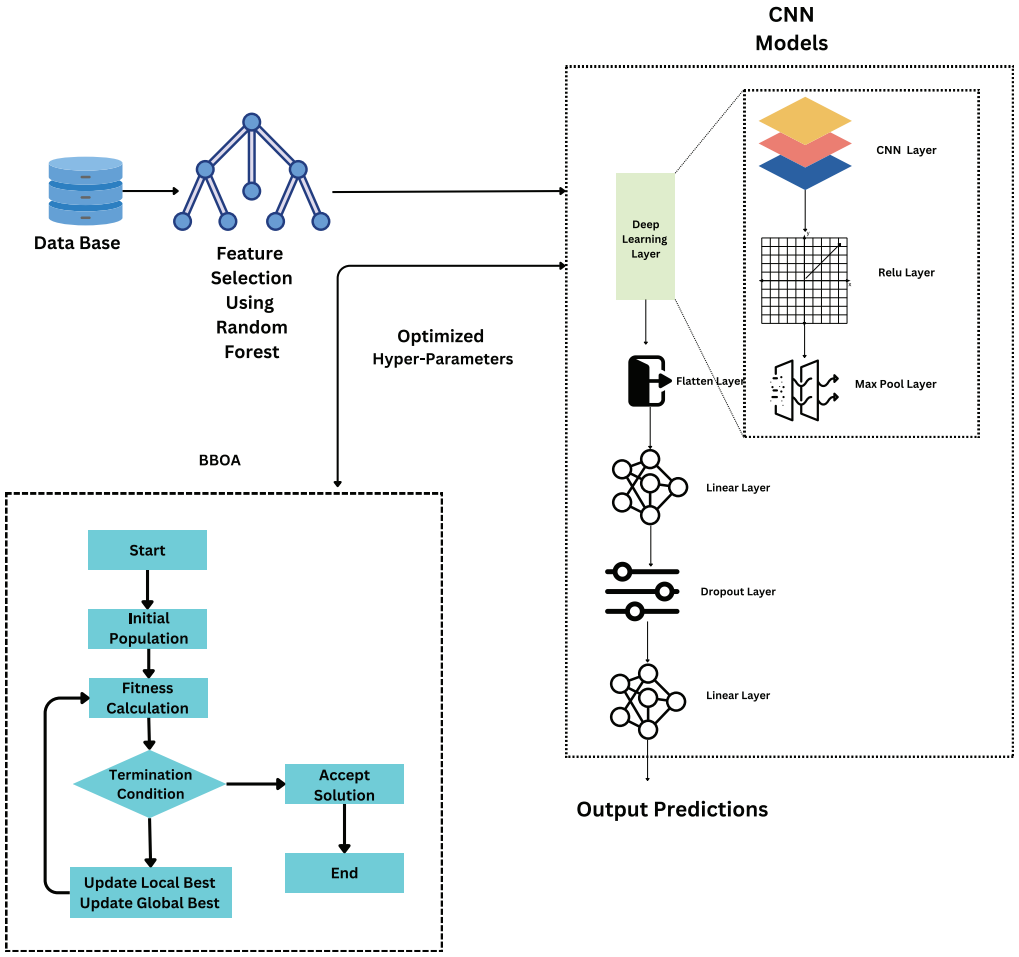

Figure 3: Proposed model

3.1 Dataset Representation and Normalization

Assume we have a dataset D with N samples, each having n features and belonging to one of k classes:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \tag{1}$$

where $x_i \in \mathbb{R}^n$ is the feature vector for sample i, and $y_i \in \{1, 2, \ldots, k\}$ is the class label.

3.2 Normalization

We apply min-max normalization to each feature to scale them to the range [0, 1]:

$$x'_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{2}$$

The normalized feature matrix X' is represented as:

$$X' = \begin{bmatrix} x'_{11} & x'_{12} & \cdots & x'_{1n} \\ x'_{21} & x'_{22} & \cdots & x'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x'_{N1} & x'_{N2} & \cdots & x'_{Nn} \end{bmatrix} \tag{3}$$

3.3 Feature Selection Using Random Forests

Random Forest is used to evaluate the importance of each feature by constructing multiple decision trees.

3.3.1 Feature Importance

For each feature f_j, its importance $I(f_j)$ is calculated as the average decrease in impurity over all trees:

$$I(f_j) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\text{nodes split on } f_j} \Delta I_t(f_j) \tag{4}$$

where T is the total number of trees.

3.3.2 Feature Selection

We select the top d features based on importance scores:

$$F = \{f_1, f_2, \ldots, f_d\} \subset \{1, 2, \ldots, n\} \tag{5}$$

The reduced feature matrix is:

$$X'_F = X'[:, F] \in \mathbb{R}^{N \times d} \tag{6}$$

3.4 CNN Model Architecture

The CNN consists of a single convolutional layer and a fully connected layer.

3.4.1 Convolutional Layer

The convolutional layer applies filters W with biases b:

$$Z = \text{ReLU}\left(X'_F * W + b\right) \tag{7}$$

where $*$ denotes the convolution operation, ReLU is the activation function.

3.4.2 Fully Connected Layer

The output of the convolutional layer is flattened and passed through a dense layer:

$$\text{Flatten}(Z) = z_1, z_2, \ldots, z_m \tag{8}$$

$$O = \sigma\left(Z \cdot W_{\text{fc}} + b_{\text{fc}}\right) \tag{9}$$

where W_{fc} and b_{fc} are the weights and biases of the fully connected layer, and σ is the softmax activation function:

$$\sigma(o_i) = \frac{e^{o_i}}{\sum_{j=1}^{k} e^{o_j}} \tag{10}$$

3.4.3 Loss Function

The cross-entropy loss is used to train the model:

$$L(Y, \hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{k} y_{ic}\log(\hat{y}_{ic}) \tag{11}$$

where Y is the true label matrix and $\hat{Y}$ is the predicted probability distribution.

3.5 Brown-Bear Optimization Algorithm (BBOA)

The BBOA is employed to optimize the CNN parameters by simulating bear behavior.

3.5.1 Initialization

Initialize a population of bears, each with random parameter vectors:

$$\theta_i = \{W_i, b_i, W_{\text{fc},i}, b_{\text{fc},i}\} \tag{12}$$

3.5.2 Fitness Evaluation

Evaluate the fitness of each bear using the loss function:

$$f(\theta_i) = L(\theta_i; X'_F, Y) \tag{13}$$

3.5.3 Bear Movement and Update

Update each bear's position using exploration and exploitation strategies:

$$\theta_i^{t+1} = \theta_i^t + \alpha \cdot \text{rand}() \cdot (\theta_{\text{best}}^t - \theta_i^t) + \beta \cdot \text{rand}() \cdot (\theta_{\text{random}}^t - \theta_i^t) \tag{14}$$

where α and β are constants, rand() generates a random number, θ_{best}^t is the best solution at iteration t, and θ_{random}^t is a randomly chosen solution.

3.5.4 Selection and Iteration

Select bears with better fitness for the next iteration. Repeat until convergence or a stopping criterion is met.

3.6 Convergence

BBOA determines convergence by monitoring the stability of the global best fitness value across iterations. Convergence is achieved when global best fitness value changes become negligible over a predefined number of iterations, indicating that the algorithm is likely near a local or global optimum. This study tracked changes over five consecutive iterations to ensure efficient convergence.

3.7 Parameter Selection

We conducted preliminary tests for parameter selection to determine optimal values for critical parameters like population size and the number of iterations. These values balance computational cost with performance, allowing the model sufficient exploration without high computational overhead. The chosen population size and iteration count provide a comprehensive solution space search, thus supporting robust optimization.

3.8 Model Evaluation

Evaluate the final model using the optimized parameters θ^* on the test dataset X_{test}:

$$\hat{Y}_{\text{test}} = \text{CNN}(X_{\text{test}}; \theta^*) \tag{15}$$

Compute performance metrics such as accuracy, precision, recall, and F1-score.

This hybrid framework effectively combines feature selection, CNN, and BBOA to enhance the detection of URL phishing attacks. Using Random Forest for feature selection reduces dimensionality, while BBOA optimizes CNN parameters for improved classification performance.

3.9 Advantage of BBOA

Compared with other optimization techniques, BBOA demonstrates several advantages. Particle Swarm Optimization (PSO), while effective for global optimization, often suffers from premature convergence in complex landscapes due to its limited exploitation capability. In contrast, BBOA's adaptive balance between exploration and exploitation allows it to navigate effectively in high-dimensional tasks. It is well-suited for optimizing complex models like CNNs in phishing detection. Genetic Algorithms (GA), on the other hand, offer a robust exploration phase but generally require a high number of generations to converge, resulting in higher computational demands. BBOA's adaptive behavior achieves efficient convergence with reduced risk of overfitting specific solutions, thereby improving the model's generalizability.

4 Result and Discussion

In this section, we provide details about the simulation environment and simulation results.

4.1 Experimental Setup

We tested our model on Windows 11, i5 system. In addition to that, we used NVIDIA GeForce RTX 3050 GPU for the simulation purpose.

We built the CNN model with the help of Pytorch [26,27], which has version 2.2.1. In addition of that, we used Pandas [28] of version 1.5.3, NumPy [29,30] of version 1.24.3, scikit-learn [31,32] of version 1.4.1.post1, and mealpy [33] of version 3.0.1.

4.2 Dataset Representation

Our proposed model, used the Kaggle data to test our proposed modelaset [34]. The dataset consists of the URLs of 11,000+ websites. In addition to that, the dataset also consists of the 30 features that are extracted from the website URL. The target variable has two classes: "Safe" and "Phishing." The distribution of the classes of the target variable is presented in Fig. 4.

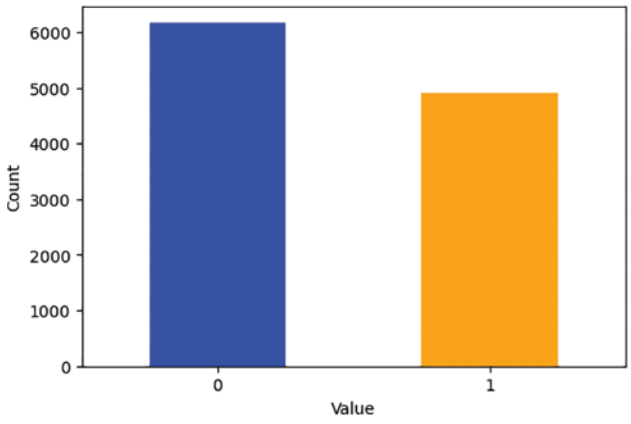

Figure 4: Class distribution

4.3 Feature Selection

As we analyze, the dataset has many features, and not every feature greatly impacts the classification of the URLs. Hence, we used a random forest-based approach for the feature selection. The details of the approach are given in Section 3. Random forest ranks the features according to the accuracy, and then the top ten highly ranked features are selected. Fig. 5 represents the rank of the features. Fig. 5 presents the rank of the feature; hence, the top ten most important features are: *'HTTPS', 'AnchorURL', 'WebsiteTraffic', 'SubDomains', 'LinksInScriptTags', 'PrefixSuffix-', 'RequestURL', 'LinksPointingToPage', 'ServerFormHandler', 'DomainRegLen'*.

We plot the correlation matrix, as represented in Fig. 6 to get more information about the ten essential features. We calculate the relation between each feature and the target variable from the correlation matrix. Table 1 presents the correlation between each feature and target variable. The table and figure help us to find the relation between the features and the target variable.

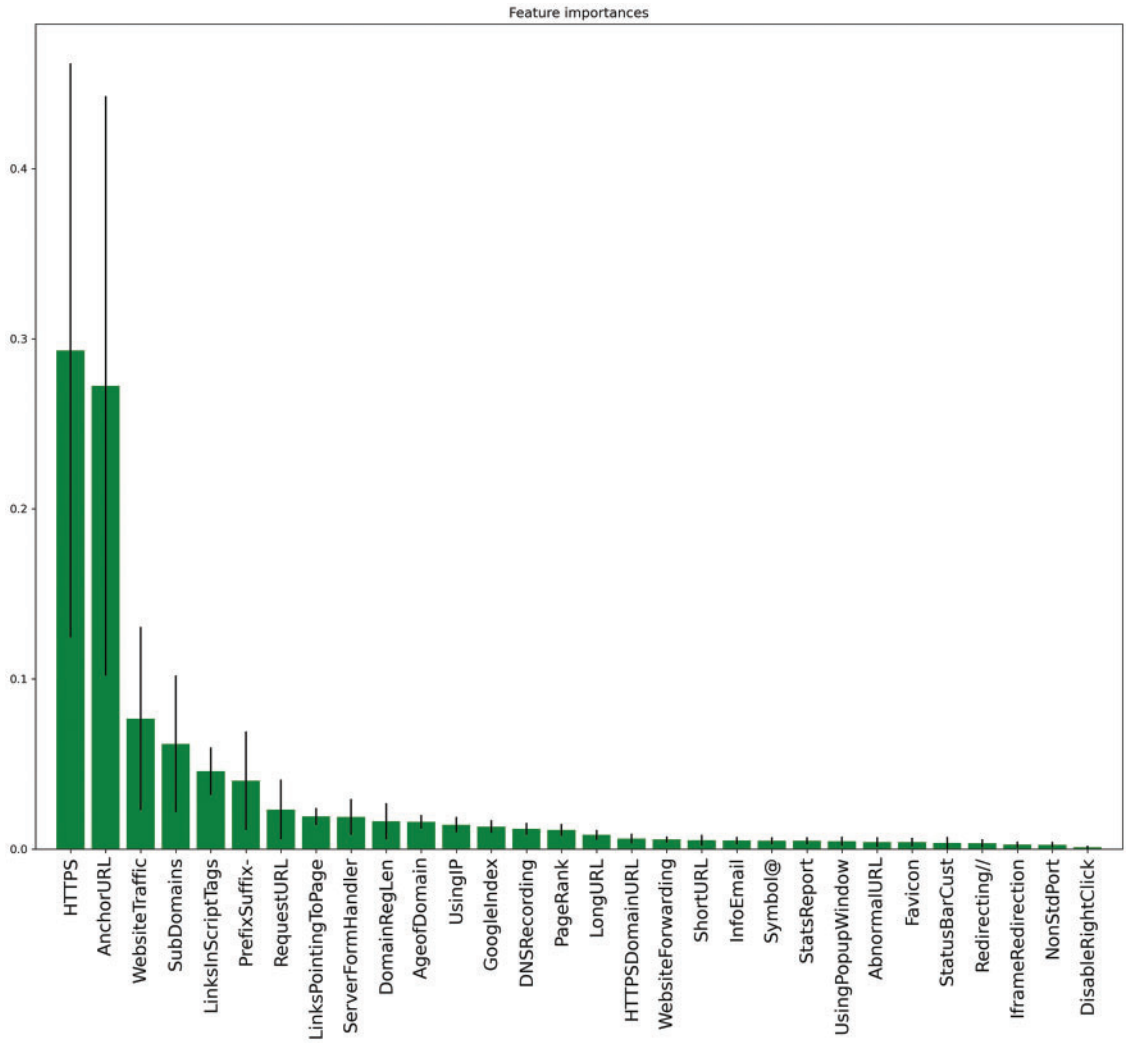

Figure 5: Feature ranking

4.4 Hyper-Parameter Tuning

After finding the relation between the features and the target classes, we tune the hyper-parameters of our CNN model with the help of Brown-Bear Optimization (BBOA). We selected the BBOA because it is designed to handle complex optimization problems with potentially rugged landscapes, similar to how brown bears navigate complex terrains in search of food. This makes it robust in finding global optima in problems where other algorithms might get stuck in local optima.

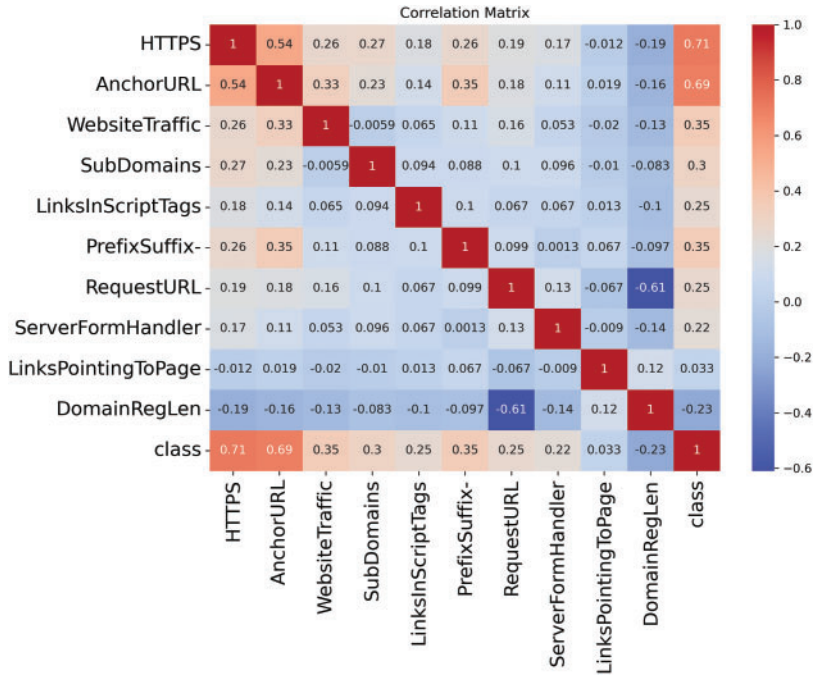

Figure 6: Correlation matrix

Table 1: Correlation between features and target

Feature	Correlation with target
HTTPS	0.714704
AnchorURL	0.692895
WebsiteTraffic	0.346003
SubDomains	0.298231
LinksInScriptTags	0.248415
PrefixSuffix-	0.348588
RequestURL	0.253478
ServerFormHandler	0.221380
LinksPointingToPage	0.032694
DomainRegLen	−0.225879

Fig. 7 presents the variation of global fitness value over the ten epochs. As represented in the figure, the value of global fitness decreases with an increase in iteration. This shows that BBOA selects the optimal values of hyper-parameters that lead to lower global fitness values. Fig. 8 presents the variation of time ichoosingng optimal hyper-parameters for each iteration. As represented in Fig. 8, the runtime decreases initially, possibly due to optimizations on population size or diversity. However, there's an increase in later iterations, which might be due to increased calculations as the algorithm attempts to escape local optima or due to more intensive exploitation processes. In addition to that, Fig. 9 presents the relation between exploration and exploitation. According to the figure, the high

exploration percentage suggests that the BBOA maintains a significant focus on discovering new areas of the search space. The relatively low and slightly declining exploitation percentage indicates a lesser emphasis on intensively searching around the best solutions found, which might be a strategy to avoid early convergence on suboptimal solutions. Finally, Fig. 10 presents the relation between diversity and the number of iterations. Fig. 10 shows a decrease in diversity from the start, stabilizing around the middle of the iterations, followed by a slight increase towards the end. This pattern suggests that the BBOA initially explores broadly, converges around certain areas, and finally expands slightly again. This figure presents the efficiency of BBOA in the detection of optimal hyper-parameters. After the process, we get the optimal value of learning rate (0.0125882) and dropout rate (0.16436758).

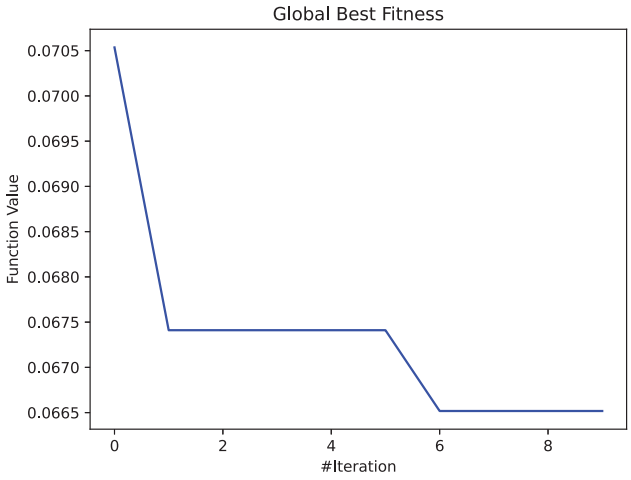

Figure 7: Global best fitness

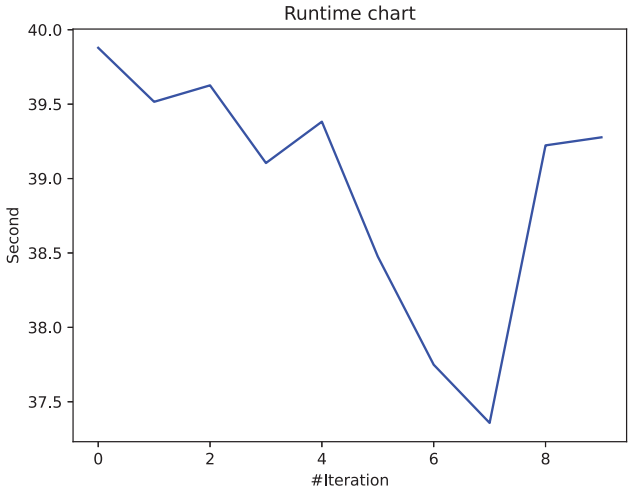

Figure 8: BBOS run time chart

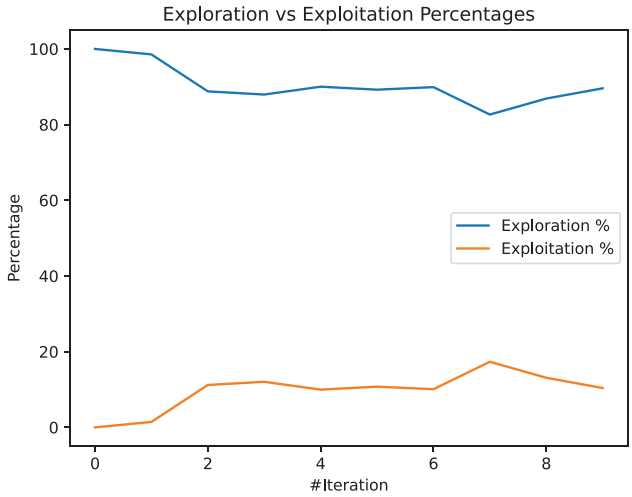

Figure 9: Exploration and exploitation chart

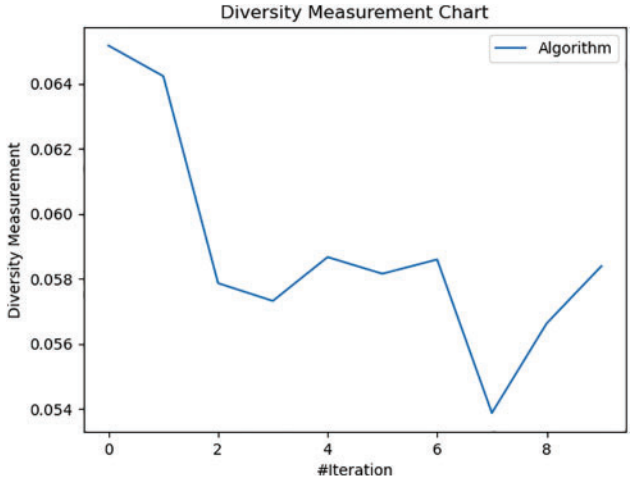

Figure 10: Diversity measurement

4.5 Performance Analysis

We used the optimal features given by the BBOA and ran the proposed CNN model for five epochs. We used the Adma optimizer and Cross Entropy Loss function in our model. We train our model for five epochs to make the proposed model lightweight. To analyze the performance of our proposed model over the epochs, we plot the accuracy and loss values over the epochs.

Fig. 11 presents the variation of training and testing accuracy over the five epochs. At the start of the epochs, the training accuracy is 0.880877, and the test accuracy is 0.921875. However, as the epochs increased, they became 0.917757, 0.920926, 0.924301, and 0.924311 in the second, third, fourth, and fifth epochs, respectively. Similarly, the testing accuracy is also increased from 0.921875 to 0.926786 in five epochs. This variation represents that our model is trained efficiently for seen and unseen data. In addition of that the convergence of training and testing accuracy concludes that the model is free from over-fitting and under-fitting.

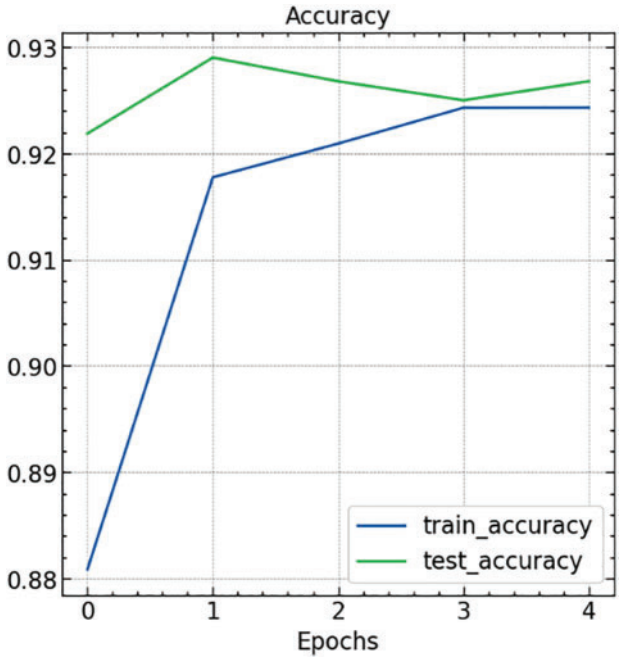

Figure 11: Accuracy variation

Apart from the acy, so plot the loss our model'slues four model's the tanning del; Fig. 12 presents this variation. The initial loss of training and test loops are 0.282105 and 0.186743, respectively, and as the number of epochs increased, the loss values for training and test decreased. The test loss decreased from 0.186743 to 0.174799, 0.175726, 0.180836, and 0.189991 in the second, third, fourth, and fifth epochs. This shows that our proposed model is performed efficiently for the unseen data values. In addition, the convergence of testing and training loss values indicated that the model is free from over-fitting and under-fitting.

Apart from the accuracy and loss, we also plot the classification report and confusion matrix, as represented in Figs. 13 and 14. From Fig. 13, it is clear that our proposed model detects the "Safe" and "Phishing" URLs with a precision of 0.92 and 0.93, respectively. In addition to that, the recall and F1-score values for the "Phishing" class are 0.94 and 0.93. That represents the effectiveness of our proposal in the detection of malicious URLs. Similarly, the recall and F1-score for the "Safe" URLs are 0.91 and 0.92, close to the "Phishing" UR values. Hence, our proposed model is not biased toward any of the classes. Finally, Fig. 13 presents that the accuracy of our proposed model is 0.93, which represents the efficiency of our proposed model.

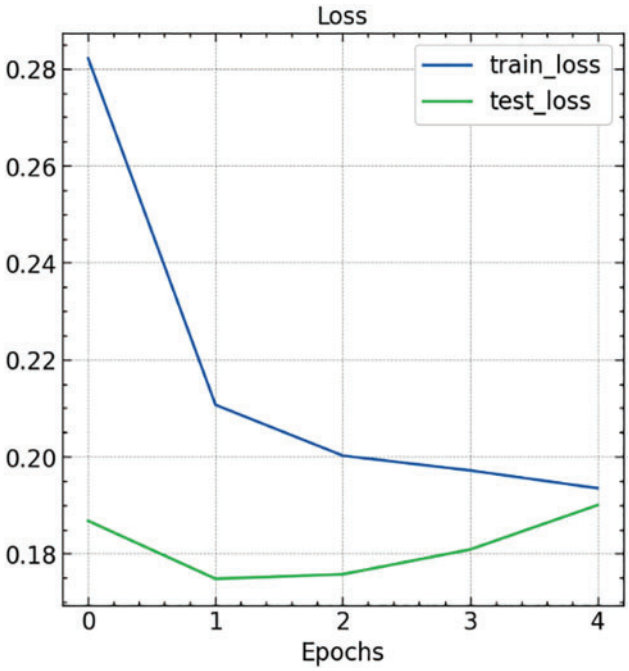

Figure 12: Loss variation

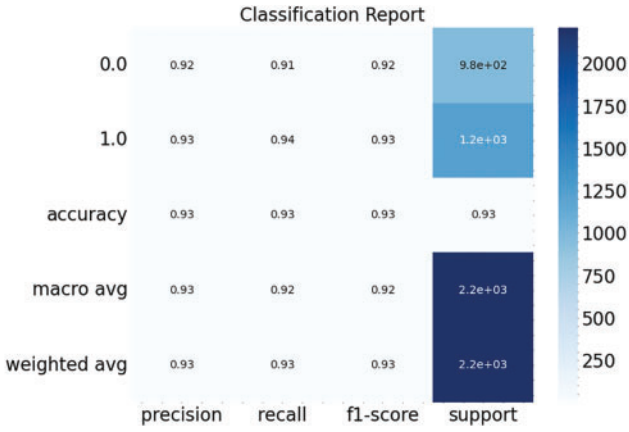

Figure 13: Classification report

Fig. 14 presents the confusion matrix of our proposed model. Fig. 14 detects 886 "Safe" URLs but cannot detect 90 URLs. Similarly, for the "Phishing" class, 1161 URLs are detected correctly, and only 90 are not correctly detected. This figure helps us find our model's actual positive and negative values. Fig. 13 shows the scope for improvement in our model, and we will improve the model to reduce the true negative values.

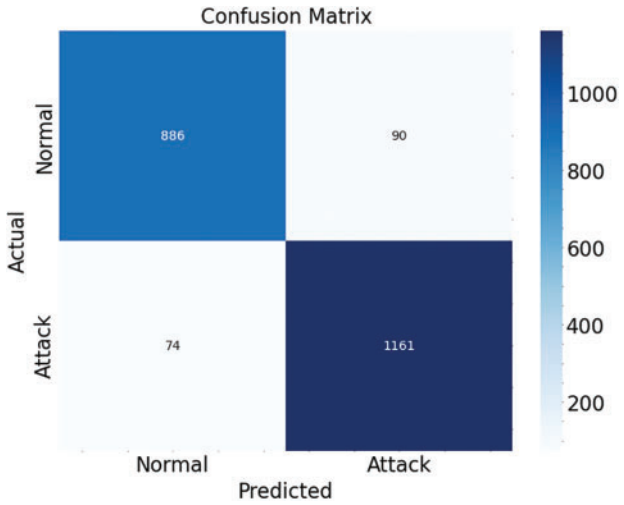

Figure 14: Confusion matrix

4.6 Comparative Analysis

We compare our proposed model with other standard ML/DL techniques, such as GRU, LSTM, RNN, ANN, SVM, and LR, to test the performance of our proposed approach. Figs. 15 and 16 present the accuracy and loss comparison ML/DL techniques with our proposed model.

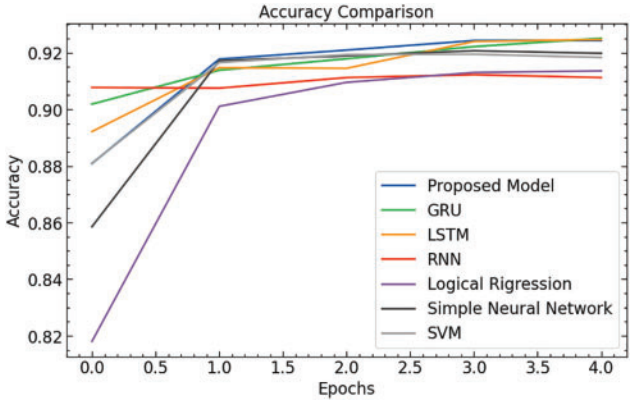

Figure 15: Accuracy comparison

According to Fig. 15 presents an accuracy comparison of GRU, LSTM, RNN, ANN, SVM, and LR across multiple epochs. The Proposed Model, indicated in the blue color line, maintains high accuracy, stabilizing just above 0.90, indicating robust performance throughout the epochs. This suggests an effective capture of underlying patterns without significant over-fitting.

In contrast, GRU and LSTM, both recurrent neural networks suitable for sequence data, initially perform lower but show rapid improvement, eventually reaching accuracies close to the Proposed Model. This highlights their effective learning capabilities over time. The standard RNN, shown in the red color line, shows less improvement over the epochs, suggesting it might be less adept at handling this task's complexities than GRU and LSTM.

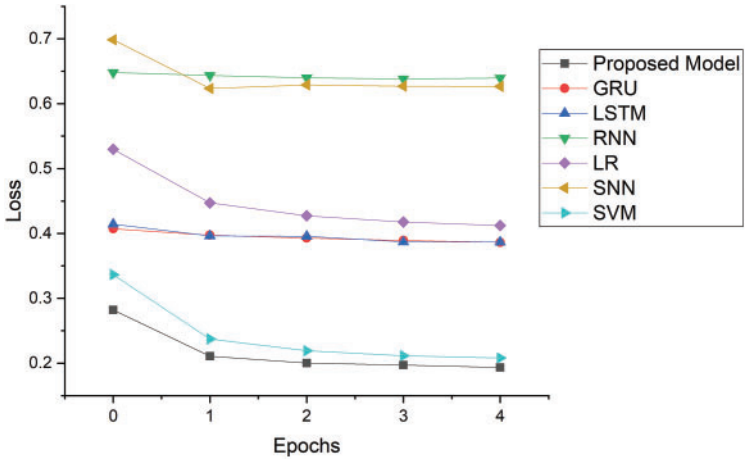

Figure 16: Loss comparison

The Simple Neural Network displays excellent and consistent performance in a grey color line, proving that simpler architectures can still be effective. The SVM model demonstrates steady accuracy, typical for non-iterative models like SVMs, though it does not improve significantly, possibly indicating a mismatch with the task complexity or dataset. Logistic Regression, mistakenly labeled as "Logical Regression" and shown in purple, begins with the lowest accuracy and declines after a slight improvement, likely due to underfitting or the model's inability to handle the dataset's complexity. Overall, this graph underscores the importance of selecting appropriate models based on dataset specifics and task requirements, with the Proposed Model showing the best overall performance.

Fig. 16 presents a loss comparison across multiple epochs. The Proposed Model, depicted in the blue line color, exhibits a decrease in loss as training progresses, reaching the lowest loss among all models. This suggests an effective learning process with efficient error minimization, potentially indicating superior handling of the dataset complexities.

The GRU and LSTM models, shown in green and orange color lines, respectively, demonstrate moderate loss values that decrease slightly over time and stabilize. Their performances are relatively close, which is characteristic of their capacity to capture temporal dependencies, albeit not as effectively as the proposed model.

However, RNN maintains a higher and more static loss than its GRU and LSTM counterparts, indicating less effective learning and potential difficulties with the task's complexities. In addition, logistic regression and the simple neural network show similar trends with a steady decrease but do not achieve loss levels as low as the proposed model.

Fig. 17 presents the comparison of time taken to train and test different models. The results in Fig. 17 show that because of their lesser complexity, classic machine learning models, including Logistic Regression and SVM, are quicker; nonetheless, this simplicity often comes at the expense of poorer accuracy in demanding activities like phishing detection. Deep learning models like CNN, GRU, and LSTM better handle complex patterns in data, which these models cannot grasp. Although our proposed CNN-BBOA model balances computational time and much greater accuracy owing to its robust feature extraction and optimal hyperparameter tuning, it is somewhat slower than more straightforward machine learning methods. This makes it appropriate for real-time applications where performance and quickness are important.

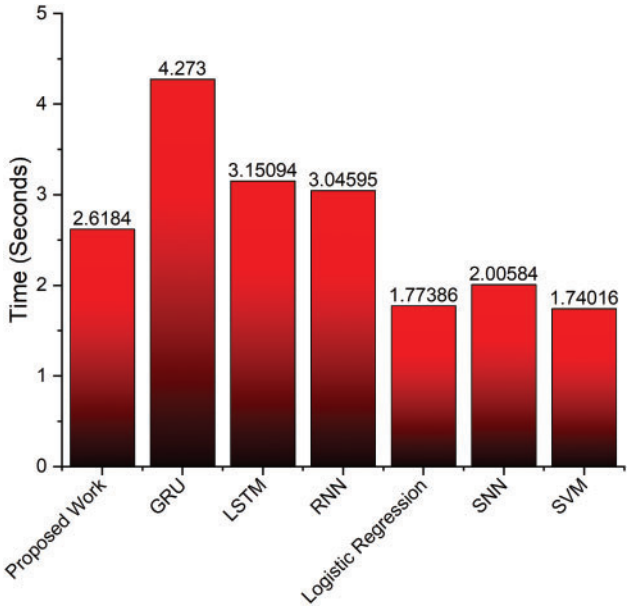

Figure 17: Running time comparison

Overall, Figs. 15–17 present the effectiveness of our proposed model as compared to the standard ML/DL techniques.

4.7 Qualitative Comparison

Table 2 presents the comparative analysis of our proposed approach with recent relevant work. The complexity of the model proposed by [7] is high because this genetic algorithm selects the best feature selection and the optimization algorithm is used in features. In the model proposed by dithm is used. The complexity of [2] is high because of the Squeeznet model. Due to the use of grid search and genetic algorithms, the model proposed by [15] is considered heavyweight. The feature selection model used by [15] used TreeSHAP, which suffers from high computational cost. The feature extract model proposed by [11] is insufficient to capture high-frequency details in datasois the complexity increased. The proposed approach by [19] used the bat algorithm; however, the bat algorithm can suffer from premature convergence in complex search spaces, which decreases the performance of the proposed approach and increases the complexity.

Table 2: Comparative analysis with past research

Model	Technique	Feature selection	Hyper-parameter tuning	Complexity
[7]	CNN	Genetic algorithm	×	High
[14]	Deep auto encoders	×	×	Moderate
[2]	SqueezeNet	SqueezeNet	FDHPO	High

(Continued)

Table 2 (continued)

Model	Technique	Feature selection	Hyper-parameter tuning	Complexity
[15]	LSTM, ANN, CNN	×	Grid search, genetic algorithm	High
[18]	Naive Bayes, random forest, XGBoost	TreeSHAP, Information Gain	×	High
[11]	DNN	VAE	×	Medium
[19]	CNN	×	BAT algorithm	High
[20]	RF, GB, and CATB	UFS	×	High
[21]	Machine learning	NLP	×	Medium
[22]	Deep learning	✓	×	Moderate
[23]	Machine learning	×	×	Moderate
[24]	Deep learning	✓	×	Moderate
[25]	Deep learning, Machine learning	✓	×	Moderate
Proposed	**CNN**	**Random forest**	**BBOA**	**Low**

Adane et al. [20] used UFS for feature selection; however, UFS evaluates each feature independently, which can miss important interactions between features and may not perform well with features that are only effective when combined with others. Due to this limitation, the efficiency of the proposed model decreased, and complexity increased. Sahingoz et al. [21] proposed an NLP-based feature extract model that extracts features for seven machine learning models. However, NLP-based feature extraction can struggle with understanding context and nuances in language, reducing effectiveness and increasing complexity. Saha et al. [22] used only the feature extract technique; however, the feature extraction technique is not efficient, which in turn increased the complexity of the approach. The approach proposed by [23] used the characterization of hyperlinks to detect phishing attacks; however, the approach did not use any optimization method. Which in turn increased the complexity. Yang et al. [24] used a two-phase detection method for the phishing websites. However, this two-phase model increased the complexity of the overall system. Mourtaji et al. [25] present a comparative analysis of different feature extraction models for detecting phishing URLs. Using a random forest approach for feature extraction and BBOA for optimization, our proposed model outperformed current phishing detection models.

4.8 Future Research Directions

Expanding the scope of the CNN-BBOA model beyond phishing detection, this approach has potential applications in other cybersecurity domains, such as fraud and intrusion detection. In fraud detection, the model could identify suspicious transaction patterns by adapting the feature selection and optimization strategies to financial datasets, where identifying subtle, anomalous patterns is critical. For intrusion detection, the CNN-BBOA framework could be applied to network traffic analysis to detect abnormal behaviors that signify potential threats, offering adaptability across various data types and security needs. Deployment considerations for the CNN-BBOA model include assessing latency and scalability, particularly in real-time applications where quick processing is essential. The

model's lightweight architecture, optimized through BBOA, supports faster computations, although real-time deployment might still require efficient hardware resources to minimize latency. Scalability could also be influenced by the increasing data demands in real-time settings, which may necessitate further model refinement to balance accuracy with performance. Additionally, limitations such as dataset-specific performance variations and sensitivity to data imbalance were noted; these factors may impact the model's robustness across different datasets, suggesting areas for further development, such as incorporating adaptive learning techniques or balancing strategies to enhance generalizability and resilience in diverse applications.

5 Conclusion

This paper presents a lightweight and optimized CNN-based model for detecting phishing URLs. Our proposed approach used feature selection and hyperparameter tuning to optimize the proposed model. We used random forest to select the ten best features and then BBOA to tune the CNN model's learning and dropout rates. An accuracy, precision, recall, and F1-score of 0.93, 0.92, 0.94, and 0.93, respectively, present the effectiveness of our proposed model. The comparative analysis with standard ML/DL techniques and past research work presents our contribution and the novelty of our proposed work. However, our proposed model still has some limitations, such as considerable true negative values. Hence, in the future, we will focus on improving our model and reducing these values. In addition, we will focus on testing our proposed model in a real-time environment.

Acknowledgement: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R 343), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number NBU-FFR-2024-1092-18.

Funding Statement: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R 343), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number NBU-FFR-2024-1092-18.

Author Contributions: Final manuscript revision, funding, supervision: Brij B. Gupta, Kwok Tai Chui; study conception and design, analysis and interpretation of results, methodology development: Shavi Bansal, Akshat Gaurav, Varsha Arya; data collection, draft manuscript preparation, figure and tables: Ahmed Alhomoud, Razaz Waheeb Attar. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data generated or analysed during this study are included in this published article.

Ethics Approval: This article contains no studies with human participants or animals performed by any authors.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] R. Alabdan, "Phishing attacks survey: Types, vectors, and technical approaches," *Fut. Inter.*, vol. 12, no. 10, 2020, Art. no. 168. doi: 10.3390/fi12100168.

[2] N. K. Kamble and N. Mishra, "Hybrid optimization enabled squeeze net for phishing attack detection," *Comput. Securi.*, vol. 144, no. 9, 2024, Art. no. 103901. doi: 10.1016/j.cose.2024.103901.

[3] A. K. Jain, B. B. Gupta, K. Kaur, P. Bhutani, W. Alhalabi and A. Almomani, "A content and URL analysis-based efficient approach to detect smishing SMS in intelligent systems," *Int. J. Intell. Syst.*, vol. 37, no. 12, pp. 11117–11141, 2022. doi: 10.1002/int.23035.

[4] A. Katiyar, "Social engineering phishing detection," 2023. Accessed: Sep. 30, 2024. [Online]. Available: https://insights2techinfo.com/social-engineering-phishing-detection/

[5] Statista, "Top phishing statistics for 2024: Latest figures and trends," 2024. Accessed: Jun. 30, 2024. [Online]. Available: https://www.statista.com/statistics/266155/number-of-phishing-domain-names-worldwide/

[6] G. Smith, "Number of unique phishing sites detected worldwide from 3rd quarter 2013 to 1st quarter 2024," 2024. Accessed: Jun. 30, 2024. [Online]. Available: https://www.stationx.net/phishing-statistics/

[7] S. J. Bu and H. J. Kim, "Optimized URL feature selection based on genetic-algorithm-embedded deep learning for phishing website detection," *Electronics*, vol. 11, no. 7, 2022, Art. no. 1090. doi: 10.3390/electronics11071090.

[8] M. M. Alani and H. Tawfik, "PhishNot: A cloud-based machine-learning approach to phishing URL detection," *Comput. Netw.*, vol. 218, 2022, Art. no. 109407. doi: 10.1016/j.comnet.2022.109407.

[9] V. Vajrobol, B. B. Gupta, and A. Gaurav, "Mutual information based logistic regression for phishing URL detection," *Cyber Secur. Appl.*, vol. 2, 2024, Art. no. 100044. doi: 10.1016/j.csa.2024.100044.

[10] S. N. Kee, "The impact of phishing on cloud-based systems and blockchain-based mitigations," 2024. Accessed: Sep. 30, 2024. [Online]. Available: https://insights2techinfo.com/the-impact-of-phishing-on-cloud-based-systems-and-blockchain-based-mitigations/

[11] M. K. Prabakaran, P. Meenakshi Sundaram, and A. Chandrasekar, "An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders," *IET Inform. Secur.*, vol. 17, no. 3, pp. 423–440, 2023.

[12] M. Aljabri *et al.*, "Detecting malicious URLs using machine learning techniques: Review and research directions," *IEEE Access*, vol. 10, pp. 121395–121417, 2022. doi: 10.1109/ACCESS.2022.3222307.

[13] B. B. Gupta, A. Gaurav, R. W. Attar, V. Arya, A. Alhomoud and K. T. Chui, "Optimized phishing detection with recurrent neural network and whale optimizer algorithm," *Comput. Mat. Contin.*, vol. 80, no. 3, pp. 4895–4916, 2024. doi: 10.32604/cmc.2024.050815.

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

[15] M. Almousa, T. Zhang, A. Sarrafzadeh, and M. Anwar, "Phishing website detection: How effective are deep learning-based models and hyperparameter optimization?" *Secur. Priv.*, vol. 5, no. 6, 2022, Art. no. e256.

[16] R. Liu *et al.*, "TransURL: Improving malicious URL detection with multi-layer Transformer encoding and multi-scale pyramid features," *Comput. Netw.*, vol. 253, 2024, Art. no. 110707. doi: 10.1016/j.comnet.2024.110707.

[17] R. Liu *et al.*, "PMANet: Malicious URL detection via post-trained language model guided multi-level feature attention network," *Inf. Fusion*, vol. 113, 2025, Art. no. 102638. doi: 10.1016/j.inffus.2024.102638.

[18] L. M. Rani, C. F. M. Foozy, and S. N. B. Mustafa, "Feature selection to enhance phishing website detection based on URL using machine learning techniques," *J. Soft Comput. Data Min.*, vol. 4, no. 1, pp. 30–41, 2023.

[19] P. P. Kumar, T. Jaya, and V. Rajendran, "SI-BBA–A novel phishing website detection based on Swarm intelligence with deep learning," *Mater. Today: Proc.*, vol. 80, no. 3, pp. 3129–3139, 2021. doi: 10.1016/j.matpr.2021.07.178.

[20] K. Adane, B. Beyene, and M. Abebe, "Single and hybrid-ensemble learning-based phishing website detection: Examining impacts of varied nature datasets and informative feature selection technique," *Digit. Thre.: Res. Pract.*, vol. 4, no. 3, pp. 1–27, 2023. doi: 10.1145/3611392.
[21] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert. Syst. Appl.*, vol. 117, no. 4, pp. 345–357, 2019. doi: 10.1016/j.eswa.2018.09.029.
[22] I. Saha, D. Sarma, R. Chakma, M. N. Alam, A. Sultana and S. Hossain, "Phishing attacks detection using deep learning approach," in *2020 Third Int. Conf. Smart Syst. Invent. Technol. (ICSSIT)*, 2020.
[23] A. K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 5, pp. 2015–2028, 2019. doi: 10.1007/s12652-018-0798-z.
[24] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019. doi: 10.1109/ACCESS.2019.2892066.
[25] Y. Mourtaji, M. Bouhorma, D. Alghazzawi, G. Aldabbagh, and A. Alghamdi, "Hybrid rule-based solution for phishing URL detection using convolutional neural network," *Wirel. Commun. Mob. Comput.*, vol. 2021, no. 1, pp. 1–24, 2021. doi: 10.1155/2021/8241104.
[26] S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan, *Programming with TensorFlow: Solution for Edge Computing Applications*. Cham: Springer, 2021, pp. 87–104. doi: 10.1007/978-3-030-57077-4.
[27] N. Ketkar, J. Moolayil, N. Ketkar, and J. Moolayil, "Introduction to PyTorch," in *Deep Learning with Python. Berkeley*. CA, USA: Apress, 2021, pp. 27–91. doi: 10.1007/978-1-4842-5364-9_2.
[28] J. Reback *et al.*, "pandas-dev/pandas: Pandas 1.0. 5," *Zenodo*, 2020. doi: 10.5281/zenodo.3898987.
[29] I. Idris, *NumPy: Beginner's Guide*. UK: Packt Publishing Ltd., 2015.
[30] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, Numpy, and Ipython*. USA: O'Reilly Media, Inc., 2012.
[31] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[32] O. Kramer and O. Kramer, *Machine Learning for Evolution Strategies*. Cham: Springer, 2016, pp. 45–53. doi: 10.1007/978-3-319-33383-0.
[33] N. Van Thieu and S. Mirjalili, "MEALPY: An open-source library for latest meta-heuristic algorithms in Python," *J. Syst. Archit.*, vol. 139, no. 12, 2023, Art. no. 102871. doi: 10.1016/j.sysarc.2023.102871.
[34] E. Chand, "Phishing website detector," 2023. Accessed: Jan. 30, 2024 . [Online]. Available: https://www.kaggle.com/datasets/eswarchandt/phishing-website-detector