sustainability

Article

Political Optimization Algorithm with a Hybrid Deep Learning Assisted Malicious URL Detection Model

Mohammed Aljebreen [1], Fatma S. Alrayes [2], Sumayh S. Aljameel [3] and Muhammad Kashif Saeed [4,*]

[1] Department of Computer Science, Community College, King Saud University, P.O. Box 28095, Riyadh 11437, Saudi Arabia
[2] Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[3] Saudi Aramco Cybersecurity Chair, Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia
[4] Department of Computer Science, Applied College, King Khalid University, P.O. Box 9004, Abha 62529, Saudi Arabia
* Correspondence: mksaeed@kku.edu.sa

Abstract: With the enhancement of the Internet of Things (IoT), smart cities have developed the idea of conventional urbanization. IoT networks permit distributed smart devices to collect and process data in smart city structures utilizing an open channel, the Internet. Accordingly, challenges like security, centralization, privacy (i.e., execution data poisoning and inference attacks), scalability, transparency, and verifiability restrict faster variations of smart cities. Detecting malicious URLs in an IoT environment is crucial to protect devices and the network from potential security threats. Malicious URL detection is an essential element of cybersecurity. It is established that malicious URL attacks mean large risks in smart cities, comprising financial damages, losses of personal identifications, online banking, losing data, and loss of user confidentiality in online businesses, namely e-commerce and employment of social media. Therefore, this paper concentrates on the proposal of a Political Optimization Algorithm by a Hybrid Deep Learning Assisted Malicious URL Detection and Classification for Cybersecurity (POAHDL-MDC) technique. The presented POAHDL-MDC technique identifies whether malicious URLs occur. To accomplish this, the POAHDL-MDC technique performs pre-processing to transform the data to a compatible format, and a Fast Text word embedding process is involved. For malicious URL recognition, a Hybrid Deep Learning (HDL) model integrates the features of stacked autoencoder (SAE) and bi-directional long short-term memory (Bi-LSTM). Finally, POA is exploited for optimum hyperparameter tuning of the HDL technique. The simulation values of the POAHDL-MDC approach are tested on a Malicious URL database, and the outcome exhibits an improvement of the POAHDL-MDC technique with a maximal accuracy of 99.31%.

Keywords: cybersecurity; smart city; Internet of Things Deep Learning; malicious URL; political optimizer

check for updates

Citation: Aljebreen, M.; Alrayes, F.S.; Aljameel, S.S.; Saeed, M.K. Political Optimization Algorithm with a Hybrid Deep Learning Assisted Malicious URL Detection Model. *Sustainability* **2023**, *15*, 16811. https://doi.org/10.3390/su152416811

Academic Editor: Fadi Al-Turjman

Received: 4 September 2023
Revised: 30 October 2023
Accepted: 2 November 2023
Published: 13 December 2023

1. Introduction

At present, there is a development of Internet of Things (IoT) mechanisms in sustainable smart environments [1]. The development of IoT devices has led to enhanced security vulnerabilities, creating general consumers as victims of various kinds of safety attacks by malicious Uniform Resource Locators (URLs), as any devices in a shared IoT system are dependent upon URLs [2]. Hackers often use phishing and spam to trick consumers by clicking malicious URLs, Trojans are embedded into computers, or the delicate data of victims may be leaked [1]. This malicious URL identification technology could assist users in finding malevolent URLs and stop users from malevolent URL attacks. Conventionally, studies on malicious URL recognition adopt blacklist-related techniques for detecting malicious URLs [2]. This technique has several exclusive benefits. It consists of a lower false-positive rate, has a high speed, and is easy to realize. Yet, today, domain generation algorithms (DGA) can produce thousands of diverse malicious field names on a daily basis, which could be identified effectively by classical blacklist-related approaches [3]. To detect malicious URLs, research scholars use an ML approach. However, such techniques should derive the features manually, and hackers can devise such attributes to avoid being recognized [4]. Confronted with the current complicated network, devising a more potentially malevolent URL identification method is a focus of study.

Aggressors can use vulnerable sites to execute malicious intent [5]. For instance, attackers inject cross-site scripting into susceptible sites to acquire the sensitive data of the target or execute phishing. Many solutions have been devised to identify these websites precisely. Such solutions are script-based, URL-related, and web content-related [6]. URL-based identification and content-related detection are the most used methods, while some research was performed on script-based identification. URL-related detection is a superior choice, as it can be a safe and proactive method for distinguishing machines; it can find malicious URLs before the user visits them [7]. Furthermore, identifying malicious URLs has the potential for resource-limited and real-time detection applications such as mobile and IoT devices. Different methods were recommended to find harmful content and malicious websites by extracting attributes from their URLs [8]. Many approaches depend on humans to derive features, whereas specific solutions make use of deep learning (DL) approaches for feature automation. Various sets of features have been used and derived for identifying host information features such as host sponsor and country name, domain features, namely .tk and .com, and lexical features, such as counting of the dots in the URL length and URL [9]. Hackers may utilize evasive approaches to bypass security countermeasures [10]. Hence, any attributes derived from such URLs are misleading since the aggressor could use them to conceal malevolent patterns and the malevolent intent of websites.

This research concentrates on the proposal of a Political Optimization Algorithm with a Hybrid Deep Learning Assisted Malicious URL Detection and Classification for Cybersecurity (POAHDL-MDC) technique. The presented POAHDL-MDC technique identifies whether malicious URLs occur or not. To accomplish this, the POAHDL-MDC technique follows pre-processing to transform it to a compatible format, and a Fast Text word embedding process is involved. For malicious URL detection, Hybrid DL (HDL) model integrates the features of SAE and Bi-LSTM. Finally, POA can be used for the optimal hyperparameter tuning of the HDL technique. The simulation results of the POAHDL-MDC methodology can be tested on a benchmark database. In short, the main contributions are given below.

- An automated POAHDL-MDC model comprising pre-processing, word embedding, HDL recognition, and POA-based hyperparameter tuning is proposed for malicious URL classification. To the best of our knowledge, the POAHDL-MDC methodology has never existed in other studies.
- The HDL classification method combines the strengths of SAE and BiLSTM models to improve the exactness of malicious URL classification.
- Hyperparameter optimization of the HDL model employing the POA model, utilizing cross-validation, aids in enhancing the forecast results of the HDLPOA-MDC technique for unseen data.

The rest of the paper is classified as follows. Section 2 produces related works, and Section 3 offers the proposed model. Then, Section 4 offers the result analysis, and Section 5 concludes the paper.

2. Related Works

Patgiri et al. [11] developed a new malicious URL detection method named DL and Bloom Filter (deepBF). DeepBF is obtainable twofold. The authors primarily devised a learned Bloom Filter using a 2D Bloom Filter. The authors experimentally determined the optimal non-cryptography string hash function. Afterwards, the authors devised a malicious URL recognition system utilizing DL. To find malicious URLs, the authors implemented the evolutionary CNN. Wanda and Jie [12] devised a deep learning using a new convolutional neural network (CNN) called URL Deep. Rather than utilizing classical CNNs, the authors employed Dynamic CNNs. It could allow the same signal on a similar CNN channel. URL Deep's graph was dynamically upgraded after all layers of the network were analyzed.

In [13], an enhanced DL-related phishing detection method was developed by incorporating the strengths of a deep neural network (DNN) and a variational autoencoder (VAE). In the structure presented, the VAE model automatically extracted the basic features of the raw URL by rebuilding the original input URL to enhance phishing URL detection. The aim of Angadi and Shukla's [14] study was to accumulate a list of significant attributes exploited to classify and detect malicious URLs. This study suggests lexical aspects and host-based URLs for increasing the efficacy of classifiers to detect malicious URLs. Utilizing ML classifiers called RF and AdaBoost techniques, Benign and Malicious URLs are categorized. In [15], the authors introduced a complete prototype of malicious URL detection through ML techniques. Specifically, the authors designed a technique utilizing the AdaBoost approach and tried a precise method of making Malicious URL exposure from an ML perspective.

In [16], the authors assessed many existing DL-oriented character-level-embedding approaches for malicious URL detection. The authors devised DeepURLDetect (DURLD), where raw URLs were encrypted through character-level embedding for transforming and using performance development. To capture different kinds of data in the URL, the authors utilized hidden layers in the DL structure to derive features in character level embedded and used a nonlinear activation function. Alsaedi et al. [17] targeted the enhancement of the recognition exactness of malicious URL recognition by developing and devising a cyber-threat intelligence-related malicious URL identification method through two-step ensemble learning. This study introduced a two-step ensemble-learning approach that combined the RF technique to pre-classify with multilayer perceptron (MLP) for decision-making.

While recent DL models have received significant attention in cybersecurity, the application of metaheuristic algorithms for optimizing hyperparameters of DL methods needs to be explored further. There is a great need to fine-tune hyperparameter values of DL models in cybersecurity tasks, which are computationally intensive and require substantial computational resources. The use of metaheuristics can optimize the DL models to eliminate the human trial and error approach. Addressing these study gaps can lead to the development of more effective and efficient DL-based cybersecurity solutions that are fine-tuned using metaheuristic algorithms, ultimately enhancing overall security posture in an increasingly digital and connected world. Some of the recently developed metaheuristic algorithms are the number hummingbird algorithm (AHA), atom search optimization, sine cosine algorithm (SCA), equilibrium optimizer (EO), the Giant Trevally Optimizer, and the Remora Optimization technique.

3. The Proposed Model

In this study, we present a unique POAHDL-MDC method for programmed recognition and classification of malicious URLs. The POAHDL-MDC approach has several stages of operations, namely pre-processing, Fast Text word embedding, HDL-based malicious URL detection, and POA-based hyperparameter tuning. Figure 1 signifies the workflow of the POAHDL-MDC methodology.

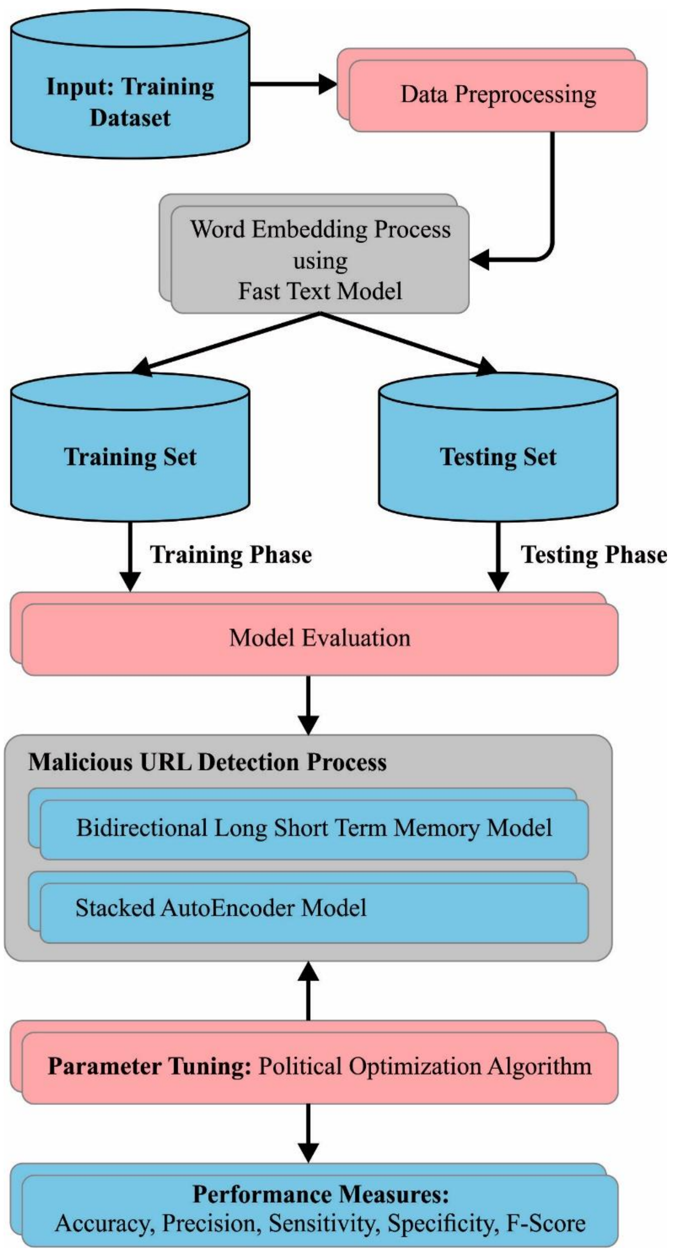
Input: Training Dataset
Data Preprocessing
Word Embedding Process using Fast Text Model
Training Set
Testing Set
Training Phase
Testing Phase
Model Evaluation
Malicious URL Detection Process
Bidirectional Long Short Term Memory Model
Stacked AutoEncoder Model
Parameter Tuning: Political Optimization Algorithm
Performance Measures: Accuracy, Precision, Sensitivity, Specificity, F-Score

Figure 1. Workflow of the POAHDL-MDC approach.

3.1. Pre-Processing

In this stage, with the help of the natural language processing (NLP) text pre-processing method, the URL is pre-processed by eradicating symbols. As URLs are crawled from websites, unnecessary texts such as punctuation, HTML codes, and symbols are eliminated to enhance classifier performance and minimize feature complexity. The gathered text data are transformed to lowercase and normalized. The normalization procedure is twofold. Initially, the text in the unstructured dataset is transformed into a structured word vector. Then, the feature vector scarcity is diminished by eliminating unwanted words and words decreased by rooting words to their original form. The normalization begins with tokenization, after the elimination of stemming, stop words, and lemmatization. Lastly, the words are transformed to their corresponding numerical formats. Stemming is a transforming procedure that converts the words into their roots, for instance, eradicating "ing" from the word and "s" from the plural words. Lemmatization converts the words using a lexical knowledge base into the base form by rooting verbs, for example, 'took' to 'take'.

3.2. Word Embedding Using Fast Text

In this work, the Fast Text technique is employed for the word embedding process. 'Word embedded' refers to a distributional representation of words, but all the words are mapped to a shared lower dimension space, and all the words are connected to a d-dimension vector [18]. In various word embedding, fastText does not ignore the word morphology. This approach is dependent upon continuous skip grams. Currently, every word can be determined as a character n-gram. Yet $n = 3$, the word rapid is as follows:

$$< qu,\ qui,\ uic,\ ick,\ ck >$$

This technique maintains subword data and evaluates valid words embedded in out-of-vocabulary words. Therefore, it offers a vector to hidden words in the trained word embedding.

For learning word representation, fastText, followed by continuous skip grams established by the author, can be easier and work well with a smaller training data count. However, this model disregards the internal world infrastructure. The fastText presents various scoring functions for preserving the subword data.

To provide the word w, the group of n grams performing in w is $N_w \subset \{1,\ N\}$, whereas N denotes the dictionary size of n-grams. The vector representation Z_g is allocated to every n-gram n. Therefore, the drive scoring function develops:

$$s(w,\ c) = \sum_{n \in N_w} Z_g^T V_c \tag{1}$$

where c denotes the context word, and V_c signifies the context vector.

3.3. Malicious URL Detection Using HDL

The HDL model is employed for automated malicious URL detection. The autoencoder (AE) refers to an unsupervised neural network mechanism that learns the hidden features of an inputted dataset, names the encoding (coding) function, while applying the learned newest feature to recreate the original input dataset, and names the decoding function [19]. AE has *one* hidden layer (HL). Significantly, the input and output layers of the AE are equivalent.

The sigmoid function is applied as s_{f^1} and sf^2, where $1 = [x_{11}, x_{12}, \ldots, x_{1dl}]^T \in R^{ld1}, b_1 \in R^{ld1}, x_2 = [x_{21}, x_{22}, \ldots, x_{2dl}]^T \in R^{2dr}, b_2 \in R^{2dr} h = \left[h_1, h_2, \cdots, h_{dh}\right]^T \in R^{dh}$, where h denotes the connection vector between x_1 and $x_2; b_1$ and b_2 represent the deviation vector.

$$h = f_1(x1) = sf1(W_{1x_1} + b_1) \tag{2}$$

$$x_2 = f_2(h) = \mathrm{s_{f2}}(W_2 h + b_2) \tag{3}$$

$$J(W, b) = J(w1, w2, b_1, b_2) = \sum_{i=1}^{N} \|x_2 - x_1\|/2N = \sum_{i=1}^{N} \|g_\theta(x_2) - x_1\|/2N \tag{4}$$

SAE represents the superposition of more than one *AEs*. Once the initial AE is implemented, successive *AEs* are implemented in order until the N-th, and the resultant output is the SAE superimposition outcome. Equation (7) signifies the variable that all AE disseminates to the following layer.

LSTM is a common kind of recurrent neural network (RNN) and is better suited for modeling time-series data, namely humidity, day-to-day air temperature, seawater salinity, air pressure, and other data attained by text buoys due to their design characteristics. In recent times, a new NN, named LSTM, has been implemented. The three major arithmetical structures in LSTM define that it achieves LSTM based on RNN.

The forgetting door is a way of selecting forget, and is given as follows:

$$f_t = \sigma\Big(W_f \cdot [h_{t-1}, x_t] + b_f\Big) \tag{5}$$

where f_t denotes outcome attained by forgetting gates, and W_f shows the vector that defines the input weight; b_f represents the bias vector; h_{t-1} indicates the HL at the final moment; the present input x_i; σ denotes the activation function:

$$\begin{aligned} W_f \cdot [h_{t-1}, x_i] = [W_f] \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \\ = \Big[W_{fh} W_{fx}\Big] \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} = W_{fh} h_{t-1} + W_{fx} x_t \end{aligned} \tag{6}$$

The input gate chooses the data that must be memorized, and it can be represented as follows:

$$\begin{cases} i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ c_t = f_t \cdot c_{t-1} + i_t \cdot \tan h(W_c \cdot [h_{t-1}, x_t] + b_c) \end{cases} \tag{7}$$

where h_{t-1} denotes resultant output at the final moment. I_t denotes the value of the input gate, c_t and c_{t-1} show the activation and cell state at the final moment, W_i represents weight in the input gate; *and* W_c denotes the forget gate's weight. b_i shows the input gate's bias vector; b_c represents the forget gate's bias vector.

The output gate can be represented as:

$$\begin{cases} o_t = \sigma(W_0 \cdot [h_{t-1}, x_t] + b_o) \\ h_t = 0_t \cdot \tan h(c_t) \end{cases} \tag{8}$$

In Equation (8), h_t represents the outcome of the output gate, O_t denotes the vector, and b_o shows the offset vector. W_o indicates the weights.

LSTM predicts the outcome at a later time, depending on the timing data of the previous time. For certain issues, the present production is relevant to the prior and future states. The principles of LSTM linking two networks remain unchanged. The forward LSTM obtains the previous dataset of input series, and the backward LSTM obtains the future dataset of input:

$$\begin{cases} \overset{\rightarrow}{h_{rf}} = LS\vec{T}M(W_1 h_{t-1}, W_2 x_t, c_{t-1}) \\ \overset{\rightarrow}{h_{tb}} = LS\vec{T}M(W_3 h_{t+1}, W_4 x_t, c_{t+1}) \\ H_t\left[\overset{\rightarrow}{h_{rf}}, \overset{\rightarrow}{h_{tb}}\right] \end{cases} \tag{9}$$

The hidden layer H_t of BLSTM at t time involves forward h_{rf} and backward h_{tb}; W_1, W_2, W_3 and W_4 are correspondingly the represent weight coefficients; x_t shows the input at t time; ht denotes the hidden state at time t.

The data transmission process accomplishes the fusion of two approaches in the HDL model: a partially supervised fine-tuning network, presenting the evaluation index, E_o, and fine-tuning the weight over the backpropagation technique, especially SAE-implemented unsupervised learning and supervised fine-tuning. In the trained method, the input dataset is mapped towards the HL over the first layer AE using Equations (2)–(4). Then, the AE is superimposed, and the whole network is well-trained until the final AE. The fine-tuning of the whole model by Equation (10) is implemented by applying backpropagation (BP) to attain a better weight.

$$E_o = \frac{1}{2} \sum_{i=1}^{N_i} (A_i - F_i)/N \tag{10}$$

where N characterizes the number of samples, A_i shows the actual value, and F_i indicates the forecasted value. Based on the SAE output, training the BLSTM network makes predictions for the prediction, training, and testing groups. The outcome can be attained afterwards by passing the comparison of the assessment conditions.

3.4. Hyperparameter Tuning

At the final stage, a POA is employed for optimum hyperparameter tuning of the HDL technique. The POA is a novel meta-heuristic system motivated by political processes like constituency allocation, party formation, party switching, inter-party elections, election campaigns, and government development [20]. POA includes five stages, given below. The party formation and constituency allotment stages take place when the population is initialized, and the residual stages are initialized to run in the loop.

The search agent in the POA includes n political parties as shown in Equation (11), where all the parties (pr_i) have n members, as shown in Equation (12). p_i^{rj} refers to the j-th members of i-th party, which can be treated as a candidate solution where p_i^{rj} denotes a vector of length d as shown in Equation (13), where d represents the number of decision variables belonging to the optimizer problems. Consequently, the size of populations is the square of n, as shown in Equation (14). Also, n constituencies exist, as shown in Equation (15). The j-th members in each party contest the election from the j-th constituencies C_j, as modeled by Equation (16).

$$pr = \{pr_1, pr_2, pr_3, \ldots,\ pr_n\} \tag{11}$$

$$pr_i = \left\{pr_i^1, pr_i^2, pr_i^3,\ \ldots, pr_i^n\right\} \tag{12}$$

$$\mathrm{pr}_i^{\mathrm{j}} = \left[pr_{i,1}^j, \cdot pr_{i,2}^j, pr_{i,3}^j, pr_{i,d}^j\right]^T \tag{13}$$

$$population\ Size = n^2 \tag{14}$$

$$Co = \{Co_1,\ Co_2,\ Co_3, \ldots,\ Co_n\} \tag{15}$$

$$Co_j = \left\{\mathrm{pr}_1^j, pr_2^{\mathrm{j}}., pr_3^j,\ pr_n^j\right\} \tag{16}$$

Election demonstrates how the election procedure is simulated. The best member in every party is named leader, i-th parties are represented as pr_i^* and the set having the party leader is signified as pr^*, demonstrated in Equation (17). After the election, the constituency winner becomes a parliamentarian. The best member from all the constituencies is regarded as the constituency winner. Co^* shows the constituency winners or parliamentarians' group, whereas Co_j^* signifies the parliamentarian or winner of the j-th constituencies, as shown below.

$$pr^* = \{pr_1^*, pr_2^*, pr_3^*, \ldots, pr_n^*\} \tag{17}$$

$$Co^* = \{Co_1^*,\ Co_2^*,\ Co_3^*, \ldots,\ Co_n^*\} \tag{18}$$

In an election campaign, every candidate solution location is upgraded based on the constituency winner $\left(Co_j^*\right)$ and the party leader (pr_i^*) is allocated by applying Equations (19) and (20) according to the best candidate in the prior iteration. Once the candidate's fitness increases, Equation (19) is exploited. Otherwise, Equation (20) is used. In all scenarios, every candidate's location is firstly upgraded based on the parliamentarian Co_j^* and the party leader pr_i^*. t shows the iteration index, r denotes the random variable within $[0,\ 1]$, and m^* first possesses the value of k-th dimensions of the leader of i-th parties $pr^*_{i,k}$, then parliamentarian $co^*_{j,k}$.

$$pr^j_{i,k}(t+1) = \begin{cases} m^* + r\left(m^* - pr^j_{i,k}(t)\right) if\ pr^j_{i,k}(t-1) \\ \leq pr^j_{i,k}(t) \leq m^*\ or\ pr^j_{i,k}(t-1) \geq pr^j_{i,k}(t) \geq m^* \\ m^* + (2r-1)\left|m^* - pr^j_{i,k}(t)\right|\ if\ pr^j_{i,k}(t-1) \\ \leq m^* \leq pr^j_{i,k}(t) or\ pr^j_{i,k}(t-1) \geq m^* \geq pr^j_{i,k}(t) \\ m^* + (2r-1)\left|m^* - pr^j_{i,k}(t-1)\right|\ if\ m^* \\ \leq pr^j_{i,k}(t-1) \leq pr^j_{i,k}(t) \text{ or } m^* \geq pr^j_{i,k}(t-1) \geq pr^j_{i,k}(t) \end{cases} \tag{19}$$

$$pr^j_{i,k}(t+1) = \begin{cases} m^* + (2r-1)\left|m^* - pr^j_{i,k}(t)\right|\ if\ pr^j_{i,k}(t-1) \leq pr^j_{i,k}(t) \\ \leq m^*\ or\ pr^j_{i,k}(t-1) \geq pr^j_{i,k}(t) \geq m^* \\ pr^j_{i,k} + r\left(pr^j_{i,k}(t) - pr^j_{i,k}(t-1)\ if pr^j_{i,k}(t-1) \leq m^* \right. \\ \leq p^{f_k(\mathcal{J})}\ or\ p\ \|_k\ (r-1) \geq m^* \geq pr^j_{i,k}(t) \\ m^* + (2r-1)\left|m^* - pr^j_{i,k}(t-1)\right| if\ m^* \\ \leq pr^j_{i,k}(t-1) \leq pr^j_{i,k}(t) \text{ or } m^* \geq pr^j_{i,k}(t-1) \geq pr^j_{i,k}(t) \end{cases} \tag{20}$$

In politics, the party-switching phase takes place concurrently with the election campaign stage, but in PO, this phase takes place after the election campaign stage. A parameter called party switching rate λ may be determined, that starts with the maximal value, λ_{max}, then declines linearly to 0, where the user tunes λ_{max}. All the party members pd_ι are selected with a certain probability, λ, to be switched with an arbitrary party p_{er}, where it substitutes the minimum fit member in that party. This phase is implemented to balance exploration and exploitation.

The constituency winners, along with the party leaders, are determined after the government formation. The entire parliamentarian Co^*_j upgrades its location based on the randomly selected constituency winner Co^*_r based on Equation (21), and if this location update results in some improvement in the fitness of Co^*_j, the location and fitness of Co^*_j are upgraded. Now, a in Equation (21) is a random integer within $[0,\ 1]$. Remember, Co^*_j is upgraded to Co^*_{jnew} only if the fitness of Co^*_{jnew} is superior to the fitness of Co^*_j.

$$Co^*_{jnew} = Co^*_r + (2a-1)\left|Co^*_r - Co^*_j\right| \tag{21}$$

Fitness selection is a considerable factor influencing the behavior of the POA method. The hyperparameter selection procedure contains a solution-encoding model to measure the effectiveness of candidate solutions. In this study, POA refers to exactness as the main criterion to plan the fitness function, expressed below:

$$Fitness = max\ (P)$$

$$P = \frac{TP}{TP+FP} \tag{22}$$

where TP and FP signify true positive and false positive values, respectively.

4. Results and Discussion

The developed technique is simulated by employing the Python 3.6.5 tool. The presented method is tested on PC i5-8600k, GeForce 1050Ti 4GB, 16GB RAM, 250GB SSD, and 1TB HDD. The experimental outcome of the POAHDL-MDC methodology can be assessed by employing a Malicious URL database [21–23] comprising 651,191 URLs with four class labels, as represented in Table 1. A set of measures is utilized in order to test the classification outcomes accuracy ($accu_y$), sensitivity ($sens_y$), specificity ($spec_y$), and F-score (F_{score}).

Table 1. Details on the dataset.

Classes	Number of URLs
Benign	428,103
Defacement	96,457
Phishing	94,111
Malware Link	32,520
Total No. of URLs	**651,191**

Sensitivity: estimates the proportion of positive samples accurately categorized.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{23}$$

Specificity: scales the proportion of negative samples exactly classified.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{24}$$

Accuracy scales the proportion of correctly classified samples (positives and negatives) against total samples (number of samples classified).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{25}$$

F-score: extends the number of true positives separated by the number of true positives plus the number of false positives.

$$\text{F-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{26}$$

The confusion matrices of the POAHDL-MDC methodology on malicious URL recognition are shown in Figure 2. The outcome highlights that the POAHDL-MDC method identifies four types of malicious URLs.

In Table 2 and Figure 3, the results of the POAHDL-MDC method, with an 80:20 ratio of TR/TS sets, are displayed. The table values signify an enhanced solution of the POAHDL-MDC system. For example, with 80% of the TR set, the POAHDL-MDC techniques attain an average $accu_y$ of 98.96%, $prec_n$ of 95.75%, $sens_y$ of 95.36%, $spec_y$ of 99.12%, and an F_{score} of 95.55%. Also, with 20% of the TS set, the POAHDL-MDC algorithm gains an average $accu_y$ of 98.94%, $prec_n$ of 95.78%, $sens_y$ of 95.24%, $spec_y$ of 99.12%, and an F_{score} of 95.50%.

In Table 3 and Figure 4, the classifier results of the POAHDL-MDC method with 70:30 of TR/TS sets are displayed. The result signifies a greater result for the POAHDL-MDC technique. For example, with 70% of the TR set, the POAHDL-MDC algorithm gains an average $accu_y$ of 99.28%, $prec_n$ of 97.04%, $sens_y$ of 97.76%, $spec_y$ of 99.43%, and an F_{score} of 97.40%. Additionally, with 30% of the TS set, the POAHDL-MDC technique gains an average $accu_y$ of 99.31%, $prec_n$ of 97.21%, $sens_y$ of 97.82%, $spec_y$ of 99.45%, and an F_{score} of 97.51%.

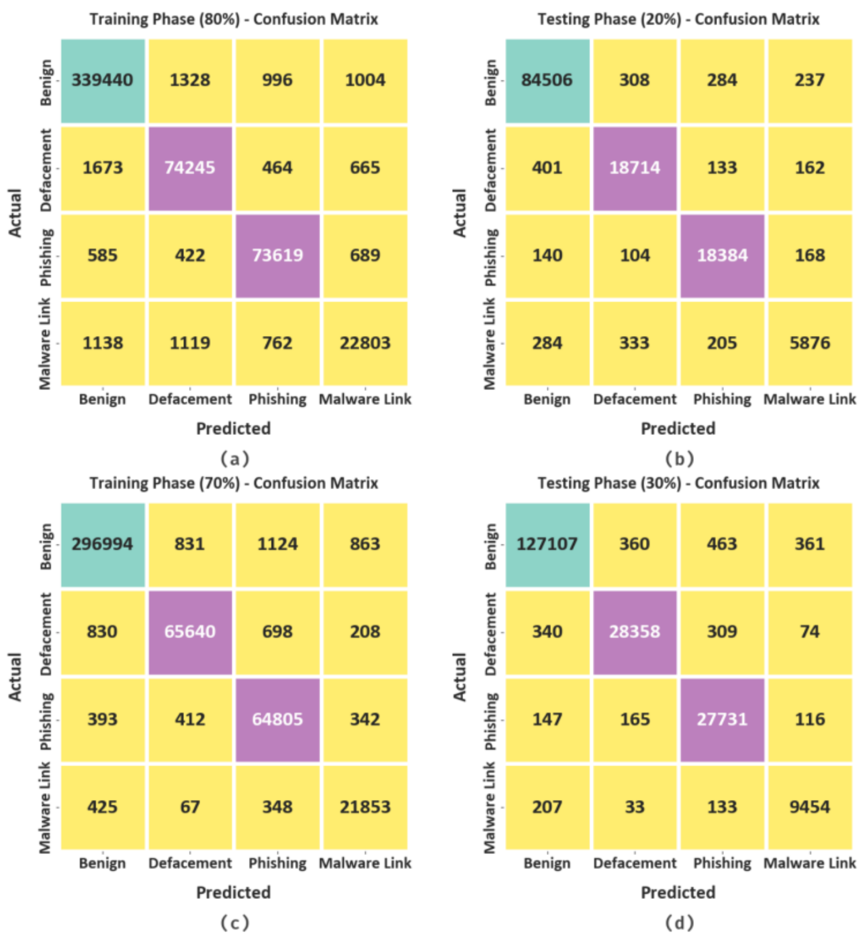

Figure 2. Confusion matrices of the POAHDL-MDC method (**a**,**b**) 80% of the TR set and 20% of the TS set and (**c**,**d**) 70% of the TR set and 30% of the TS set.

Table 2. Classifier outcome of the POAHDL-MDC method on 80% of TR set and 20% of TS set.

Class	$Accu_y$	$Prec_n$	$Sens_y$	$Spec_y$	F_{score}
		Training Phase (80%)			
Benign	98.71	99.01	99.03	98.09	99.02
Defacement	98.91	96.28	96.36	99.35	96.32
Phishing	99.25	97.07	97.75	99.50	97.41
Malware Link	98.97	90.63	88.31	99.52	89.45
Average	98.96	95.75	95.36	99.12	95.55
		Testing Phase (20%)			
Benign	98.73	99.03	99.03	98.16	99.03
Defacement	98.89	96.17	96.41	99.33	96.29
Phishing	99.21	96.73	97.81	99.44	97.26
Malware Link	98.93	91.20	87.73	99.54	89.43
Average	98.94	95.78	95.24	99.12	95.50

Figure 5 inspects the $accu_y$ of the POAHDL-MDC algorithm on the $train_g$ and val_d procedures on the test database. The result implies that the POAHDL-MDC technique gains superior $accu_y$ values above maximal epochs. Additionally, the enhanced val_d $accu_y$

over $train_g$ $accu_y$ demonstrates that the POAHDL-MDC algorithm obtains better results on the test database.

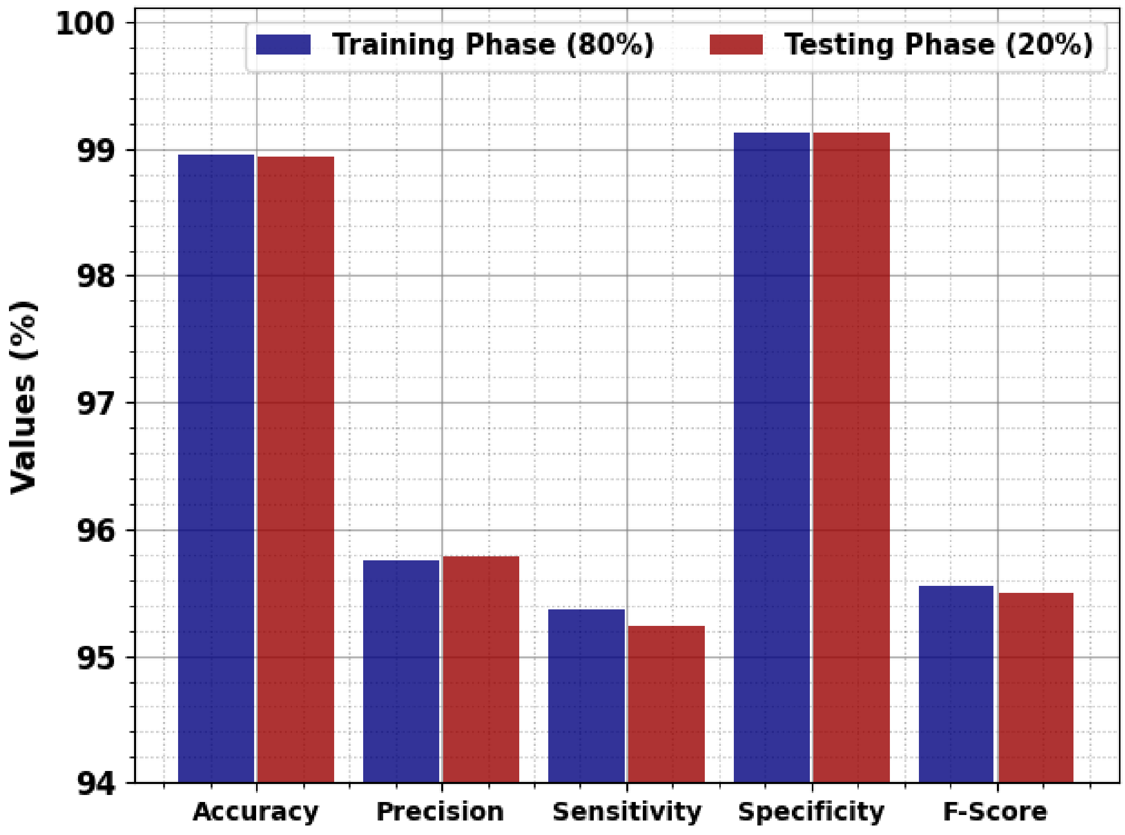

Figure 3. Classifier outcome of the POAHDL-MDC technique on 80% of the TR set and 20% of the TS set.

Table 3. Classifier result of the POAHDL-MDC model on 70% of the TR set and 30% of the TS set.

Class	$Accu_y$	$Prec_n$	$Sens_y$	$Spec_y$	F_{score}
		Training Phase (70%)			
Benign	99.02	99.45	99.06	98.94	99.25
Defacement	99.33	98.04	97.42	99.66	97.73
Phishing	99.27	96.76	98.26	99.44	97.50
Malware Link	99.51	93.93	96.30	99.67	95.10
Average	99.28	97.04	97.76	99.43	97.40
		Testing Phase (30%)			
Benign	99.04	99.46	99.08	98.97	99.27
Defacement	99.34	98.07	97.51	99.66	97.79
Phishing	99.32	96.84	98.48	99.46	97.65
Malware Link	99.53	94.49	96.20	99.70	95.34
Average	99.31	97.21	97.82	99.45	97.51

The loss curve of the POAHDL-MDC model at the time of $train_g$ and val_d is shown on the test database in Figure 6. The result represents the POAHDL-MDC approach gains nearby values of $train_g$ and val_d loss. It could be detected that the POAHDL-MDC system obtains results efficiently on the test database.

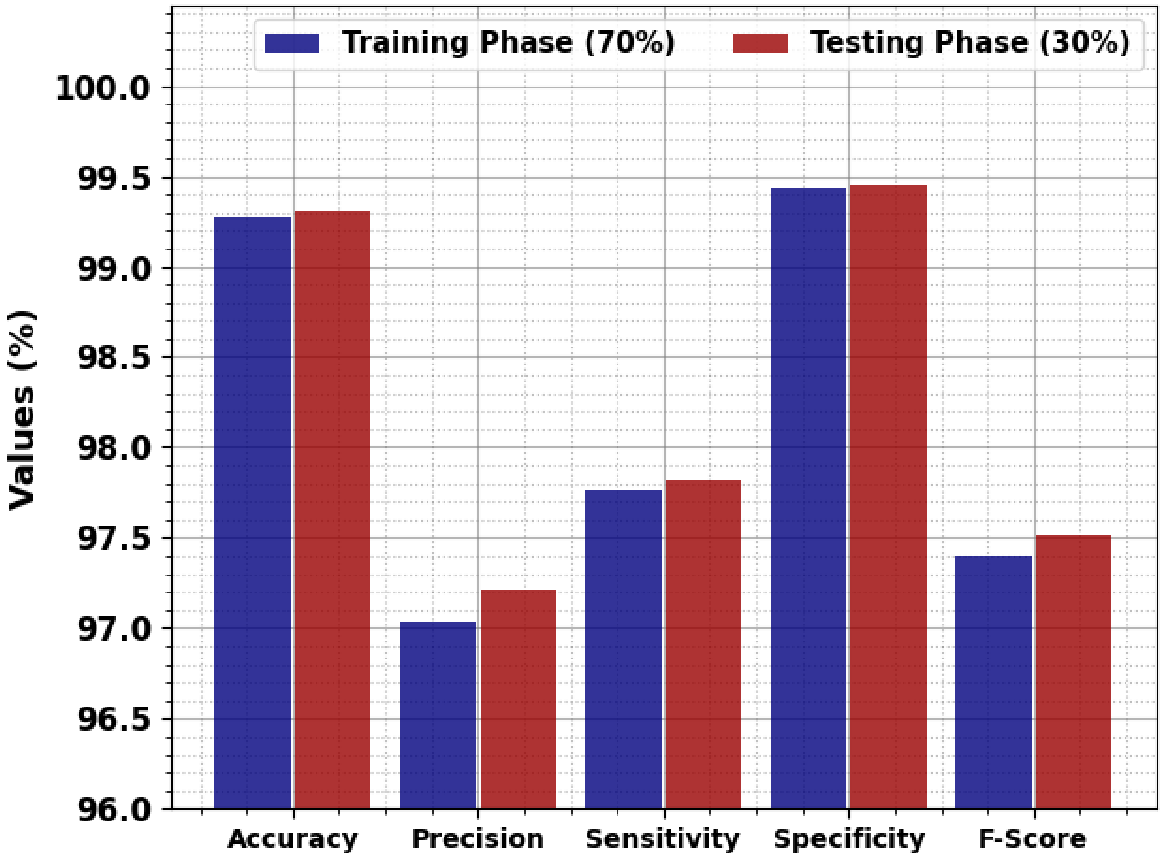

Figure 4. Classifier outcome of the POAHDL-MDC technique on 70% of the TR set and 30% of the TS set.

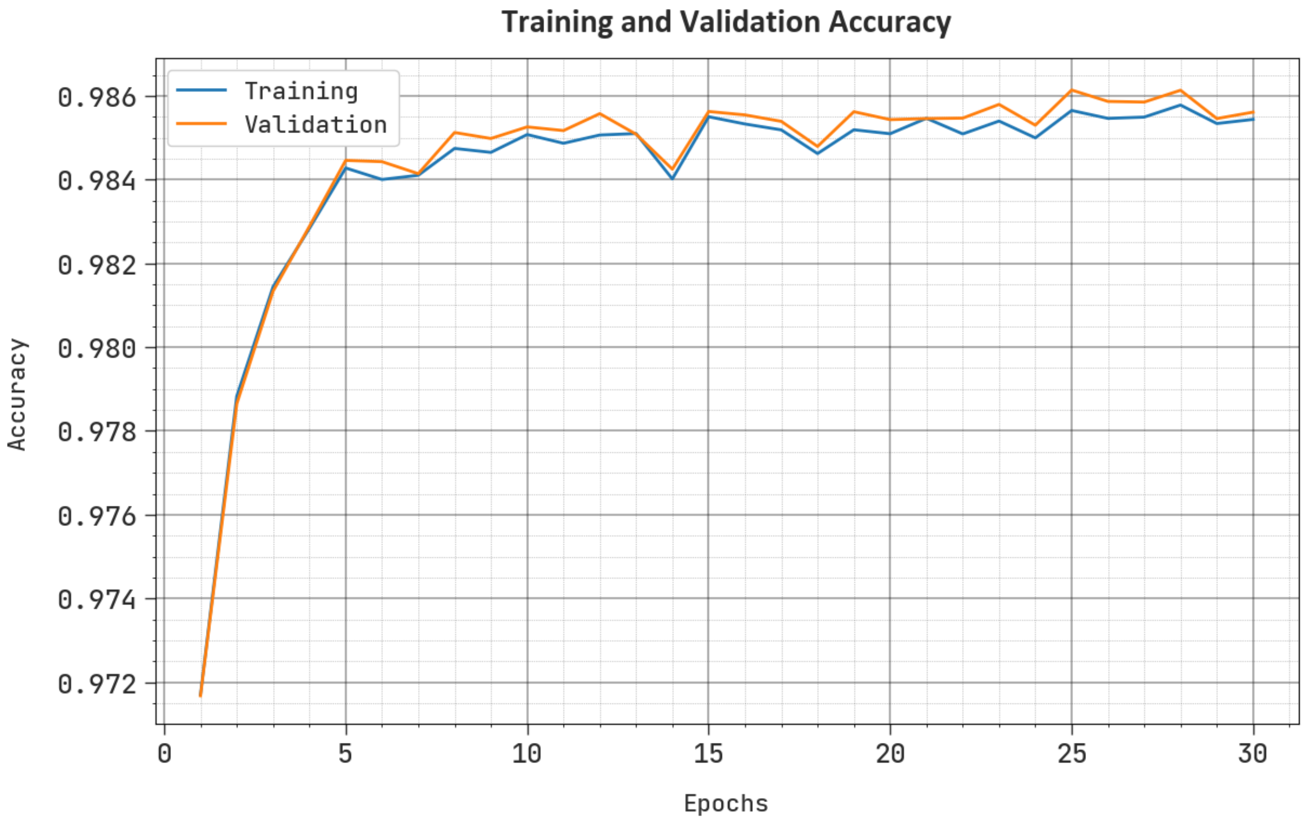

Figure 5. Accuracy curve of the POAHDL-MDC methodology.

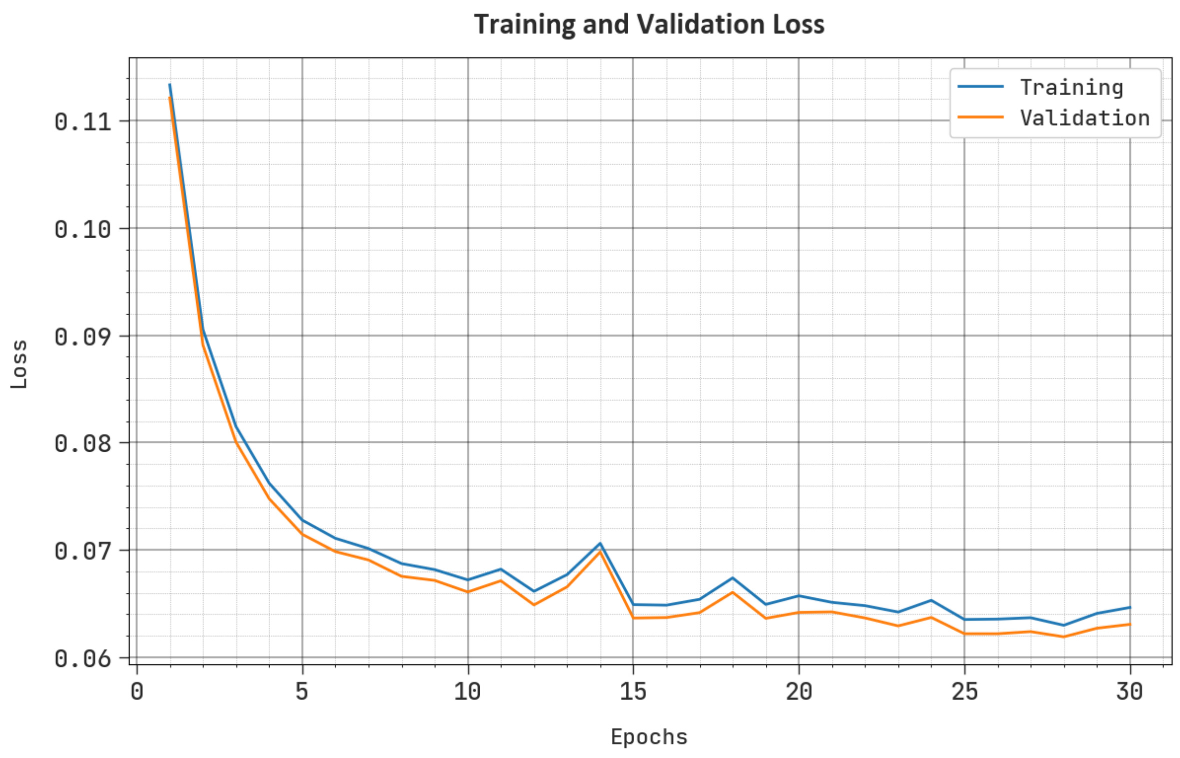

Figure 6. Loss curve of the POAHDL-MDC algorithm.

A comprehensive PR analysis of the POAHDL-MDC model applied to the test dataset is illustrated in Figure 7. The figure infers that the POAHDL-MDC system outcomes have greater values of PR. Also, the POAHDL-MDC algorithm has superior PR values in four classes.

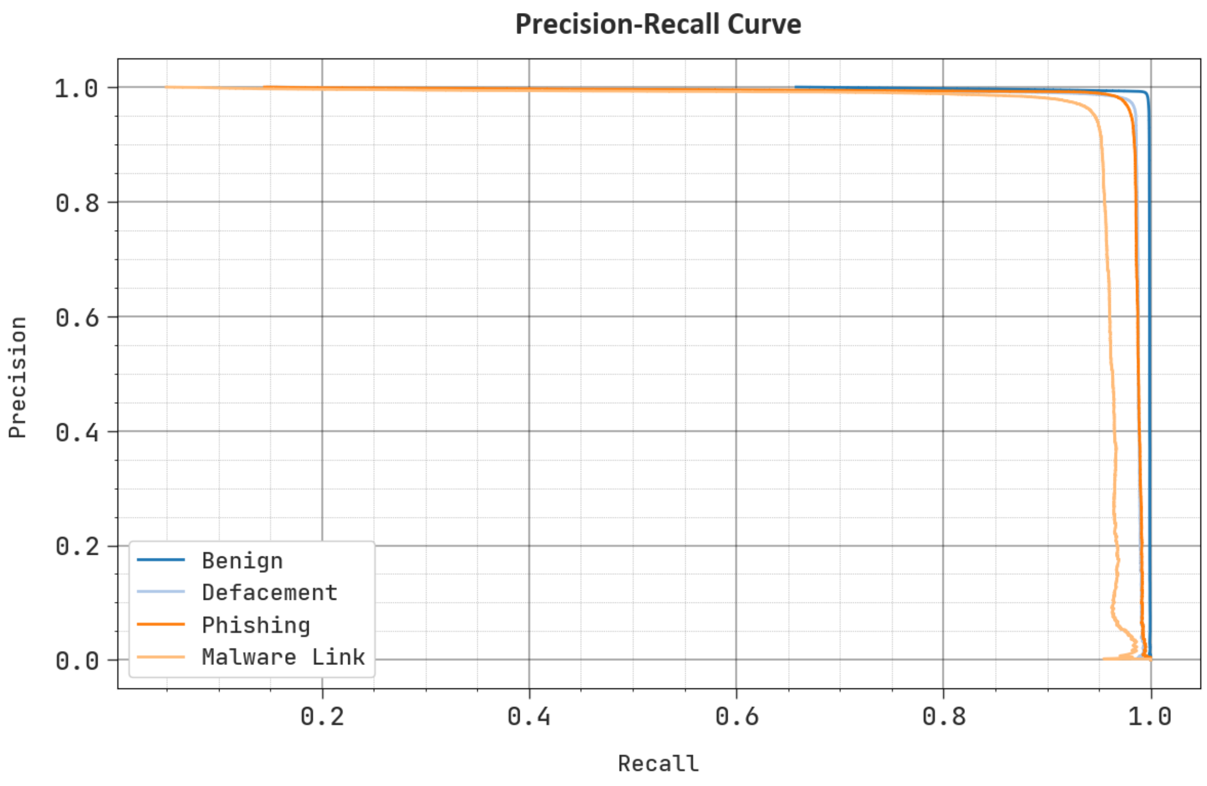

Figure 7. PR curve of the POAHDL-MDC approach.

In Figure 8, an ROC curve for the POAHDL-MDC model is revealed for the test database. The result reveals that the approach improves ROC values. Further, the POAHDL-MDC approach exhibits greater ROC values in all four classes.

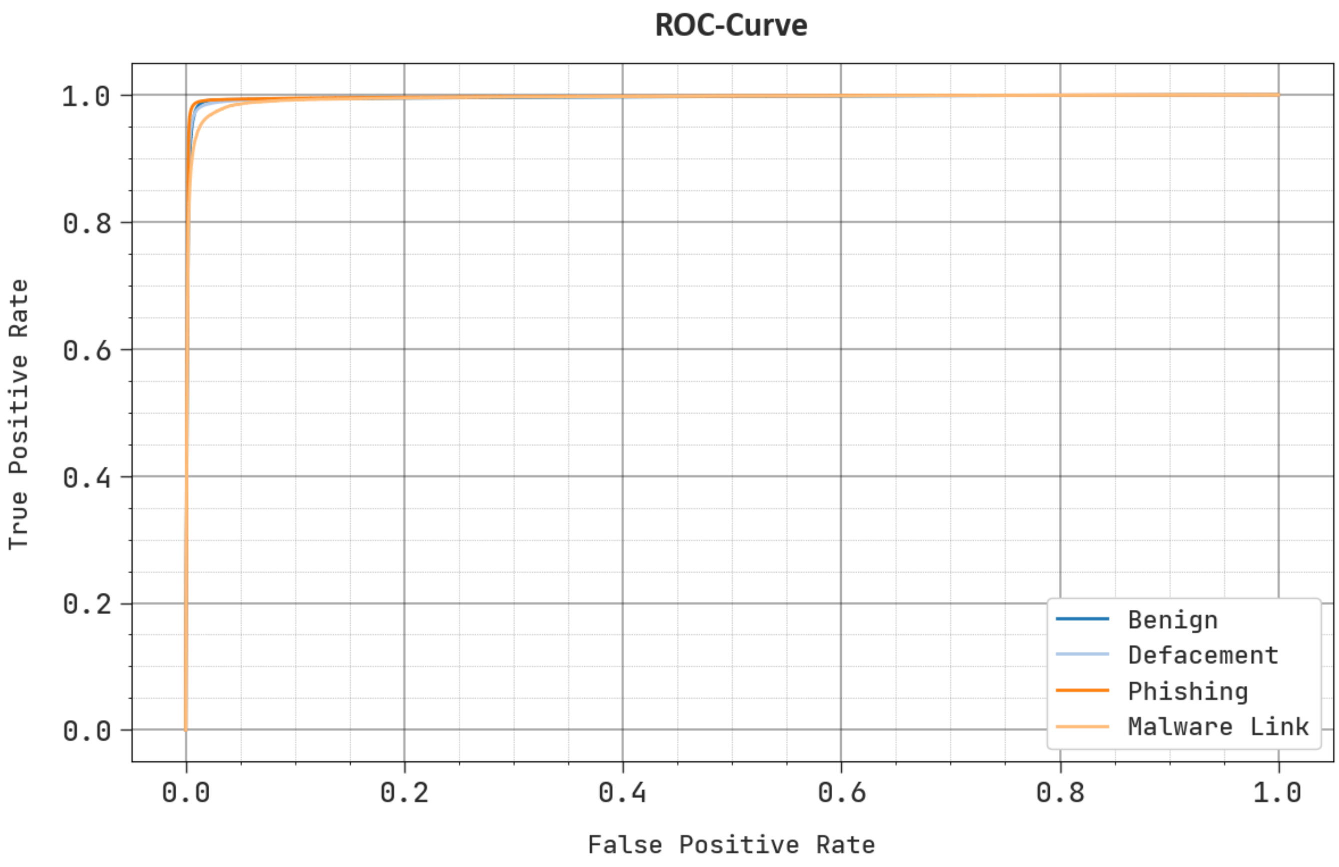

Figure 8. ROC curve of the POAHDL-MDC approach.

In Table 4 and Figure 9, a clear comparison of the POAHDL-MDC system with existing approaches is made [17]. The results highlight that the LR and RF approaches accomplish the lowest outcome.

Table 4. Comparative outcome of the POAHDL-MDC methodology with other systems [17].

Methods	$Accu_y$	$Sens_y$	$Spec_y$	F_{score}
POAHDL-MDC	99.31	97.82	99.45	97.51
Sequential DL	98.58	97.32	98.80	96.96
Naïve Bayes	98.33	94.71	97.75	94.54
Logistic Reg.	95.22	96.66	98.08	95.75
Decision Tree	98.40	95.06	95.24	94.13
Random Forest	95.33	97.31	95.23	96.56
Conv. NN	98.92	96.98	97.53	94.66

At the same time, sequential DL, NB, DT, and CNN techniques achieve closer outcomes. But the POAHDL-MDC technique gains outperforming results with a maximum $accu_y$ of 99.31%, $sens_y$ of 97.82%, $spec_y$ of 99.45%, and F_{score} of 97.51%. These outcomes confirm the superior solution of the POAHDL-MDC model over other current approaches. The improved URL detection results of the POAHDL-MDC technique are based on the inclusion of POA-based hyperparameter tuning. An application of POA selects optimum hyperparameter values of the HDL technique. Hyperparameters are not learned at the time of training but set earlier to training. They have an essential effect on the performance of the technique, as picking optimal values leads to improved exactness. By use of POA-based hyperparameter tuning, the POAHDL-MDC technique gains superior outcomes by concentrating on the most appropriate features and choosing optimal settings for the algorithm. These results guaranteed enhanced behavior of the POAHDL-MDC method when compared to existing models.

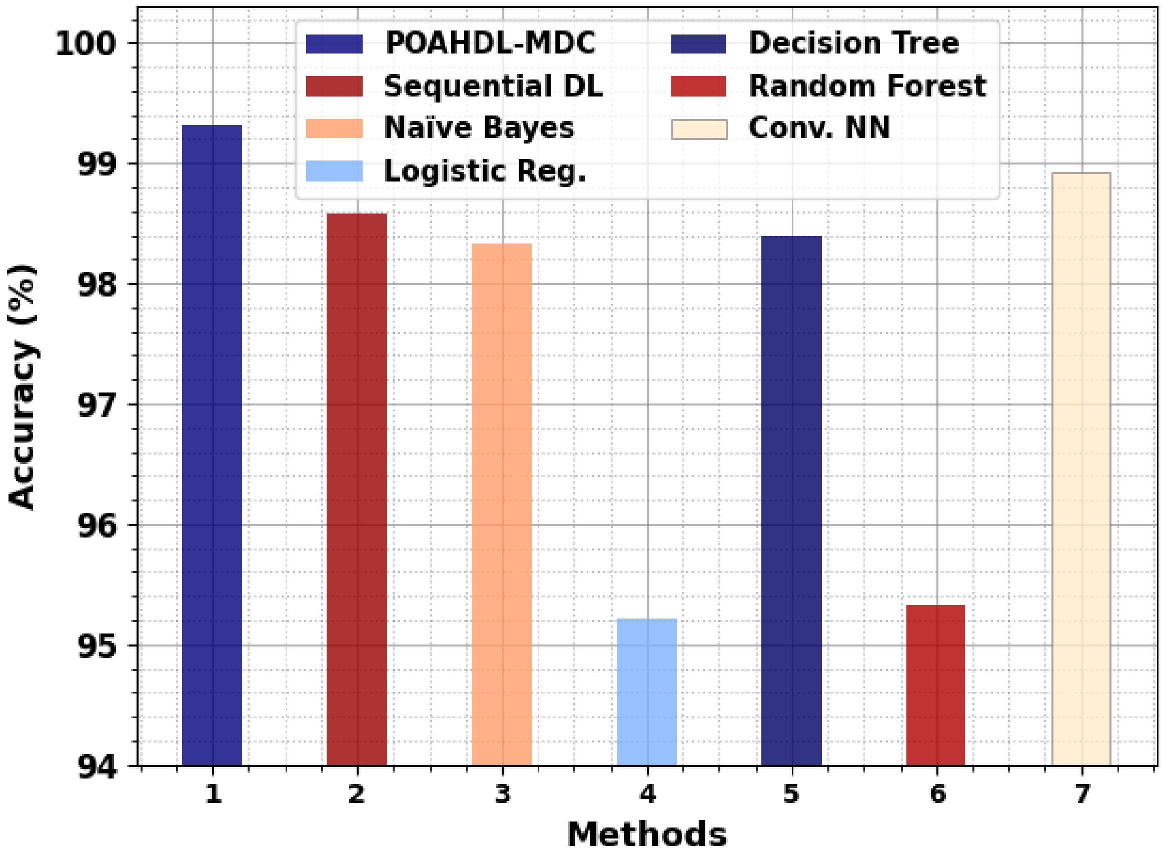

Figure 9. $Accu_y$ outcome of the POAHDL-MDC approach with existing methods.

5. Conclusions

In this study, we proposed a new POAHDL-MDC methodology for the automated recognition and classification of malicious URLs. To accomplish this, the POAHDL-MDC approach initially performed data pre-processing to change the data to a compatible format, and a Fast Text word embedding process was involved. For malicious URL detection, the HDL model integrating the features of SAE and Bi-LSTM models was utilized. Lastly, POA was employed for optimum hyperparameter tuning of the HDL methodology. The simulation value of the POAHDL-MDC technology was verified on a benchmark database, and the outcome revealed better results for the POAHDL-MDC methodology for various measures. In future, a hybrid metaheuristic-based feature selection process could be designed to reduce the high dimensionality problem and thereby enhance the detection rate. In addition, future work could examine a combination of many data modalities, such as text, network traffic, and user behavior, into DL models. In addition, new approaches such as attention-based models, graph neural networks, or transformer-based models could be used for capturing complex patterns in URLs and their associated features.

Author Contributions: Conceptualization, M.A. and S.S.A.; Methodology, F.S.A., S.S.A. and M.K.S.; Software, S.S.A.; Validation, F.S.A., S.S.A. and M.K.S.; Investigation, M.A.; Data curation, F.S.A.;

Writing–original draft, M.A., F.S.A. and M.K.S.; Writing—review & editing, S.S.A. and M.K.S.; Visualization, F.S.A.; Supervision, M.A.; Project administration, M.K.S.; Funding acquisition, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through large group Research Project under grant number (RGP2/117/44). Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R319), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. Research Supporting Project number (RSP2024R459), King Saud University, Riyadh, Saudi Arabia. We Would like to thank SAUDI ARAMCO Cybersecurity Chair for funding this project.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing does not apply to this article as no datasets were generated during the current study.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Kim, D.; Shin, J.; Seo, J.T. A Study on Log Collection to Analyze Causes of Malware Infection in IoT Devices in Smart City Environments. *J. Korean Soc. Internet Inf.* **2023**, *24*, 17–26.
2. Sundhari, R.M.; Jaikumar, K. IoT assisted Hierarchical Computation Strategic Making (HCSM) and Dynamic Stochastic Optimization Technique (DSOT) for energy optimization in wireless sensor networks for smart city monitoring. *Comput. Commun.* **2020**, *150*, 226–234. [CrossRef]
3. Contreras-Masse, R.; Ochoa-Zezzatti, A.; García, V.; Pérez-Dominguez, L.; Elizondo-Cortés, M. Implementing a novel use of multicriteria decision analysis to select IIoT platforms for smart manufacturing. *Symmetry* **2020**, *12*, 368. [CrossRef]
4. Al-Turjman, F.; Zahmatkesh, H.; Shahroze, R. An overview of security and privacy in smart cities' IoT communications. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3677. [CrossRef]
5. Kumar, N.; Goel, V.; Ranjan, R.; Altuwairiqi, M.; Alyami, H.; Asakipaam, S.A. A Blockchain-Oriented Framework for Cloud-Assisted System to Countermeasure Phishing for Establishing Secure Smart City. *Secur. Commun. Netw.* **2023**, *2023*, 8168075. [CrossRef]
6. Janet, B.; Nikam, A. Real-Time Malicious URL Detection on Twitch using Machine Learning. In Proceedings of the IEEE 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 16–18 March 2022; pp. 1185–1189.
7. Do Xuan, C.; Nguyen, H.D.; Tisenko, V.N. Malicious URL detection based on machine learning. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*.
8. Raja, A.S.; Pradeepa, G.; Arulkumar, N. Mudhr: Malicious URL detection using a heuristic rules-based approach. In Proceedings of the AIP Conference Proceedings, Krishnagiri, India, 19 May 2022; AIP Publishing LLC: Melville, NY, USA, 2022; Volume 2393, p. 020176.
9. Swarnkar, M.; Sharma, N.; Kumar Thakkar, H. Malicious URL Detection Using Machine Learning. In *Predictive Data Security using AI: Insights and Issues of Blockchain, IoT, and DevOps*; Springer Nature: Singapore, 2022; pp. 199–216.
10. Li, T.; Kou, G.; Peng, Y. Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Inf. Syst.* **2020**, *91*, 101494. [CrossRef]
11. Patgiri, R.; Biswas, A.; Nayak, S. deepBF: Malicious URL detection using learned bloom filter and evolutionary deep learning. *Comput. Commun.* **2023**, *200*, 30–41. [CrossRef]
12. Wanda, P.; Jie, H.J. URLDeep: Continuous Prediction of Malicious URL with Dynamic Deep Learning in Social Networks. *Int. J. Netw. Secur.* **2019**, *21*, 971–978.
13. Prabakaran, M.K.; Chandrasekar, A.D.; Meenakshi Sundaram, P. An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders. *IET Inf. Secur.* **2023**, *17*, 423–440. [CrossRef]
14. Angadi, S.; Shukla, S. Malicious URL Detection Using Machine Learning Techniques. In *Intelligent Sustainable Systems: Proceedings of ICISS 2022*; Springer Nature: Singapore, 2022; pp. 657–669.
15. Khan, F.; Ahamed, J.; Kadry, S.; Ramasamy, L.K. Detecting malicious URLs using binary classification through the ada boost algorithm. *Int. J. Electr. Comput. Eng. (2088–8708)* **2020**, *10*.
16. Srinivasan, S.; Vinayakumar, R.; Arunachalam, A.; Alazab, M.; Soman, K.P. DURLD: Malicious URL Detection using Deep Learning-Based Character-Level Representations. In *Malware Analysis Using Artificial Intelligence and Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 535–554.
17. Alsaedi, M.; Ghaleb, F.A.; Saeed, F.; Ahmad, J.; Alasli, M. Cyber threat intelligence-based malicious URL detection model using ensemble learning. *Sensors* **2022**, *22*, 3373. [CrossRef] [PubMed]

18. Mojumder, P.; Hasan, M.; Hossain, M.F.; Hasan, K.A. A study of fast text word embedding effects in document classification in the bangla language. In Proceedings of the Cyber Security and Computer Science: Second EAI International Conference—ICONCS 2020, Dhaka, Bangladesh, 15–16 February 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 441–453.
19. Wang, Y.; Guo, J.; Yang, Z.; Dou, Y.; Chang, X.; Sun, R.; Zuo, G.; Yang, W.; Liang, C.; Hao, Y.; et al. Computer prediction of seawater sensor parameters in the central arctic region based on hybrid machine learning algorithms. *IEEE Access* **2020**, *8*, 213783–213798. [CrossRef]
20. Askari, Q.; Younas, I.; Saeed, M. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. In *Knowledge-Based Systems*; Elsevier: Amsterdam, The Netherlands, 2020; Volume 195, p. 105709.
21. Kaggle. Malicious URLs Dataset. Available online: https://www.kaggle.com/sid321axn/malicious-urls-dataset (accessed on 3 September 2023).
22. PhishTank. Join the Fight against Phishing. Available online: https://phishtank.org/ (accessed on 3 September 2023).
23. University of New Brunswick. URL Dataset (ISCX-URL2016). Available online: https://www.unb.ca/cic/datasets/url-2016.html (accessed on 3 September 2023).