electronics

MDPI

Article

Modeling of Botnet Detection Using Barnacles Mating Optimizer with Machine Learning Model for Internet of Things Environment

Fatma S. Alrayes [1], Mohammed Maray [2], Abdulbaset Gaddah [3], Ayman Yafoz [4], Raed Alsini [4], Omar Alghushairy [5], Heba Mohsen [6] and Abdelwahed Motwakel [7,*]

[1] Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[2] Department of Information Systems, College of Computer Science, King Khalid University, Abha 61413, Saudi Arabia
[3] Department of Computer Sciences, College of Computing and Information System, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[4] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[5] Department of Information Systems and Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah 21589, Saudi Arabia
[6] Department of Computer Science, Faculty of Computers and Information Technology, Future University in Egypt, New Cairo 11835, Egypt
[7] Department of Computer and Self Development, Prince Sattam bin Abdulaziz University, AlKharj 16278, Saudi Arabia
* Correspondence: a.ismaeil@psau.edu.sa

Abstract: Owing to the development and expansion of energy-aware sensing devices and autonomous and intelligent systems, the Internet of Things (IoT) has gained remarkable growth and found uses in several day-to-day applications. However, IoT devices are highly prone to botnet attacks. To mitigate this threat, a lightweight and anomaly-based detection mechanism that can create profiles for malicious and normal actions on IoT networks could be developed. Additionally, the massive volume of data generated by IoT gadgets could be analyzed by machine learning (ML) methods. Recently, several deep learning (DL)-related mechanisms have been modeled to detect attacks on the IoT. This article designs a botnet detection model using the barnacles mating optimizer with machine learning (BND-BMOML) for the IoT environment. The presented BND-BMOML model focuses on the identification and recognition of botnets in the IoT environment. To accomplish this, the BND-BMOML model initially follows a data standardization approach. In the presented BND-BMOML model, the BMO algorithm is employed to select a useful set of features. For botnet detection, the BND-BMOML model in this study employs an Elman neural network (ENN) model. Finally, the presented BND-BMOML model uses a chicken swarm optimization (CSO) algorithm for the parameter tuning process, demonstrating the novelty of the work. The BND-BMOML method was experimentally validated using a benchmark dataset and the outcomes indicated significant improvements in performance over existing methods.

Keywords: Internet of Things; cybersecurity; botnet detection; deep learning; feature selection

Citation: S. Alrayes, F.; Maray, M.; Gaddah, A.; Yafoz, A.; Alsini, R.; Alghushairy, O.; Mohsen, H.; Motwakel, A. Modeling of Botnet Detection Using Barnacles Mating Optimizer with Machine Learning Model for Internet of Things Environment. *Electronics* **2022**, *11*, 3411. https://doi.org/10.3390/electronics11203411

Academic Editors: Thippa Reddy Gadekallu and Celestine Iwendi

Received: 28 August 2022
Accepted: 13 September 2022
Published: 21 October 2022

1. Introduction

The Internet of Things (IoT) refers to an interconnected network of software, devices, actuators, sensors, and so on that exchange and store information. The advantages of the IoT include the flow of information, automation, and communication with less effort and time [1]. In the IoT structure, physical gadgets have capacities for organization and management derived from being smart gadgets, and such gadgets can become a vigorous part of human life, ranging from the home to big institutional and industrial fields. The

IoT brings great innovation to lives by enabling indirect transmission among gadgets and individuals, making it susceptible to various cyber scams [2]. For the IoT, numerous security solutions have been devised, such as prevention, authentication, and detection. Using machine learning (ML) techniques with the IoT might resolve issues regarding privacy and security. Nowadays, it has become crucial to determine where automated techniques for rapid decision making should be run, such as the fog, the cloud, or the thin layer [3]. However, when all ML decisions are performed in the cloud, the IoT decision-making process is delayed. With other layers—namely, the fog or thin layer—it is difficult to apply ML solutions because of inadequate resources; namely, energy, bandwidth, and processing [4].

Several research scholars are trying to defend against botnet assaults on the IoT atmosphere. However, several gaps exist for the formulation of an effective detection mechanism. To deal with these attacks, an intrusion detection system (IDS) is one effective method [5]. However, conventional IDSs can often be positioned for IoT settings because of such problems. Complicated cryptographic systems could be embedded in many IoT gadgets for the usual reasons [6]. There are generally two types of IDSs: misuse and anomaly techniques. The misuse-related techniques, termed signature-related techniques, depend on the signs of attacks and are found in many public IDSs.

Existing research notes that deep learning (DL) approaches can detect IoT assaults highly efficiently compared to conventional ML techniques [7]. However, only the cloud layer has the resources for running such techniques. Moreover, such approaches are not continuously active in certain situations, such as remote live functioning, as the mechanism is supposed to constitute realistic decisions quickly [8]. Preceding work on IoT assaults has shown that an ML approach, such as support vector machine (SVM), could offer meaningful outcomes when it is linked with an optimization algorithm or feature reduction or extraction method [9]. This amalgamation of methods fails to address the low source requirement. ML approaches, such as K-nearest neighbors (KNNs), decision trees (DTs), naïve Bayes (NB), and others, are tremendously useful for applications such as non-interactive or offline predictions among small datasets [10]. Supporting all such variables, deep learning (DL) plays a significant role in the medical sector in maintaining security against numerous kinds of assaults and shedding light on the well-known ransomware attacks.

In this study, we designed a botnet detection model using barnacles mating optimizer with machine learning (BND-BMOML) for the IoT environment. For data normalization, the proposed model uses the Z-score normalization technique. Furthermore, the BMO algorithm is employed to select a useful set of features. Finally, chicken swarm optimization (CSO) with the Elman neural network (ENN) model is used for botnet detection. The experimental validation of the BND-BMOML model was carried out using a benchmark dataset.

2. Literature Review

In Vinayakumar et al. [11], a new botnet detection technique was developed on the basis of two-stage DL architecture to semantically distinguish botnet and legitimate performances at the application layer of the domain name system (DNS) service. Initially, the similarity measure of DNS queries is evaluated through a Siamese network based on predetermined thresholds for choosing the commonest DNS data across Ethernet connections. Next, a domain generation model related to DL framework is recommended for classifying normal and abnormal domain names. In [12], the presented method aims to recognize IoT botnet attacks initiated from compromised IoT gadgets by using the efficacy of the new grey wolf optimizer (GWO) algorithm to discover the features that better define IoT botnet complexity and simultaneously improve the hyperparameter of the one class support vector machine (OCSVM). Popoola et al. [13] developed a robust DL-oriented botnet attack detection technique that could manage extremely imbalanced network traffic datasets. In particular, the synthetic minority oversampling technique (SMOTE) produces further minority samples to achieve class balance, whereas DRNN learns hierarchical feature representation from the balanced network traffic dataset to implement discriminatory classification.

Sriram et al. [14] developed a DL-oriented botnet framework that functions on network traffic flow. The presented method gathers network traffic flow, transforms it to connection records, and applies a DL algorithm to identify assaults originating from the compromised IoT gadgets. Habib et al. [15] developed a detection system based on multi-objective particle swarm optimization (MOPSO) to recognize malicious behavior in IoT network traffic. The efficiency of MOPSO can be confirmed in contrast to filter-based feature selection methods, conventional ML algorithms, and the multi-objective non-dominating sorting genetic algorithm (NSGA-II). Wu et al. [16] designed a common architecture based on the deep reinforcement learning (DRL) technique that efficiently produces adversarial traffic flows to deceive the detection technique by automatically adding perturbation to the sample. During the entire process, the target detector is considered a black box and to be closer to real-time attacks. An RL agent is armed to upgrade the adversarial instances by merging the feedback from the target models (malicious or benign) and the series of activities and is capable of changing the spatial and temporal features of traffic flows when preserving the executability and original functionality.

The author of [17] addresses the IoT cybersecurity threat in smart cities and develops an anomaly detection-IoT (AD-IoT) technique using a smart anomaly detection-based random forest approach. The presented technique could efficiently identify compromised IoT devices at distributed fog nodes. McDermott et al. [18] developed a solution to the recognition of botnet activity within networks and consumer IoT gadgets. A new application of the DL technique was utilized to develop a detection method related to the bidirectional long short term memory (Bi-LSTM)-based recurrent neural network (RNN). Then, word embeddings were used for recognition of attack packets and text conversion into tokenized integer format. The proposed technique was compared with the LSTM-RNN in identifying four attack vectors utilized by the Mirai botnet and the loss and accuracy were estimated.

3. The Proposed Model

In this article, a new BND-BMOML algorithm was developed for the identification and recognition of botnets in the IoT environment. To accomplish this, the BND-BMOML model initially follows a data standardization approach. In the presented BND-BMOML model, the BMO algorithm is employed to select a useful set of features. For botnet detection, the BND-BMOML technique employs the CSO with ENN model in this study. Figure 1 demonstrates the block diagram of the BND-BMOML system.

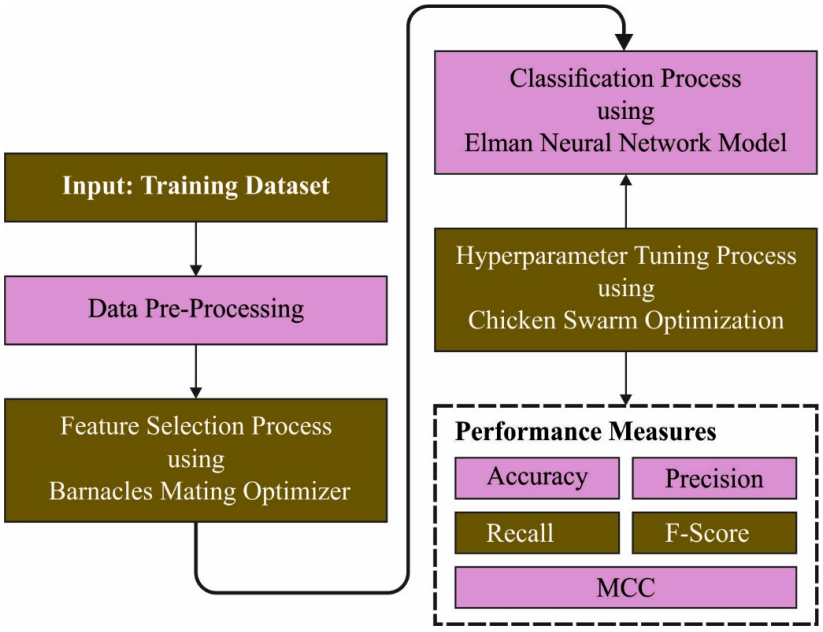

Figure 1. Block diagram of BND-BMOML approach.

3.1. Data Standardization

The data standardization procedure (DSP) is a crucial stage in data preprocessing used primarily to provide feature scaling to ensure features are on nearly similar scales, such that all the features are equivalently significant. The DSP makes the data easy to process with the ML algorithm. In the study, the standardization process (Z-score normalization) was used, where each feature is rescaled to make sure the standard deviation and mean are within the range of 0 and 1, correspondingly. In this research, the Z-score normalization was employed as follows:

$$X_{stand} = \frac{X - mean\ (X)}{Standard\ Deviation\ (X)} \tag{1}$$

The Z-score normalization is effective for different optimization approaches and, specifically, gradient descent (GD), which is widely applied by ML algorithms. The aim of standardization is to enhance the performance of ML algorithms and avoid or mitigate bias in ML classification.

3.2. Feature Selection Using BMO Algorithm

At this stage, the BMO algorithm is employed to select a useful set of features from the preprocessed data, thereby increasing accuracy and reducing computation complexity [19]. Barnacles are often found permanently attached to solid substances, such as ships, rocks, sea turtles, and corals. Barnacles are hermaphroditic organisms that have male and female reproduction systems, and the unique feature of barnacles is their penis size, which can stretch to more than the length of their body (up to seven or eight times). The barnacle mating takes place through sperm-cast and normal copulation. In the mating of isolated barnacles, sperm-cast takes place. This can be performed by discharging the fertilized eggs into water. Such behaviors of barnacles in releasing novel offspring provide insights into the use of BMO for resolving the problem of optimization. Like other evolutionary approaches, such as the genetic algorithm (GA), BMO employs the same technique to develop the selection method for the parent to be mated for producing novel offspring. However, the solution process is dissimilar from the GA and does not utilizing familiar selection techniques; namely/ tournament, roulette wheels, and so on. The selection procedure for barnacles that mate can be undertaken according the subsequent rules:

- Although barnacles are recognized as hermaphroditic organisms, female barnacles can be fertilized by one or more male barnacles, where all the barnacles are mated with each other to prevent complications;
- The value of *pl* should be initially set by the user and the selective barnacle parents can be arbitrarily performed. The value of *pl* is the control variable in these algorithms that could be tuned to attain better optimization outcomes, along with maximum iterations and number of barnacles;
- The Hardy–Weinberg principle is used when the selective barnacle parents lie in the range of *pl*. Then, the sperm-cast is imposed to achieve novel offspring.

The generation of novel offspring is guided by the Hardy–Weinberg principle as follows:

$$x_i^{N\_new} = px_{barnacle\_m}^N + qx_{barnacle\_d}^N\ for\ k \leq pl \tag{2}$$

$$x_i^{N\_new} = rand() \times x_{barnacle\_m}^N\ for\ k > pl \tag{3}$$

where $k = |barnacle\_m - barncle\_d|$, p indicates the random number uniformly distributed, $q = (1 - p)$, and $x_{barnacle\_m}^N$ and $x_{barnacle\_d}^N$ show the variables chosen randomly for the barnacle parents correspondingly. Furthermore, rand() means the random integer ranges between zero and one $(0 \sim 1)$. p and q characterize the inheritance percentage from the corresponding barnacles' parents. For instance, p is set to 0.80. This shows that the novel offspring inherit 20% (100% − 20%) of the father's features and 80% of the behavior or features of the mother. Equation (4) is used for the exploitation of optimization whereas

Equation is used as the exploration of the proposed BMO. Further, it is noteworthy that the exploration (sperm-cast) can only be related to the mother's barnacles because they receive sperm discharged from another barnacle elsewhere. As soon as the barnacle breeds, the number in the population will be doubled from the early population. To control these expansions, something has to be implemented. Like the GA, a sorting method is required in BMO where a better outcome for a specific iteration is positioned at the top half of the doubled populations.

In the modeled BMO technique, the fitness function (FF) is employed to balance the classifier accuracy (maximal) and the selected feature count in every solution (minimal) obtained with the selected feature. Equation (10) symbolizes the FF for the computing solution:

$$Fitness = \alpha\gamma_R(D) + \beta\frac{|R|}{|C|} \tag{4}$$

At this point, $\gamma_R(D)$ denotes the classifier error rate of a presented classifier (K-nearest neighbor (KNN) classifier). $|R|$ represents the cardinality of chosen set and $|C|$ indicates the number of features in the data. α and β display the two variables for the significance of classification quality and subset length, respectively. $\in[1, 0]$ and $\beta = 1 - \alpha$.

3.3. Botnet Detection Using ENN Model

In this study, the ENN model was exploited for botnet detection. The ENN consists of output, input, hidden, and context layers [20]. The major formation of this NN is FFNN; therefore, the relationship within the output, input, and hidden layers (HLs) is completely associated with the multi-layer NN. Furthermore, there exists another layer in the ENN, which is called the context layer. Its input comes from the output of HLs and it stores the initial values of the HLs. The external input, output, and context weight matrixes are defined by W_h^0, W_h^i, W_h^c. By considering the ENN form, the dimensions of the input, as well as output layers, are n—viz., $x^1(t) = [x_1^1(t), x_2^1(t), \ldots., x_n^1(t)]^T$ and $y(t) = [y_1(t),\ y_2(t),\ \ldots,\ y_n(t)]^T$—and the dimension of the context layer is m:

$$u_i(l) = e_i(l), \quad i = 1, 2,\ \ldots,\ n \tag{5}$$

In Equation (5), l indicates the output and input layers in iteration l. Then, k HLs are considered as follows:

$$\begin{gathered} v_k(l) = \sum_{j=1}^{N} \omega_{kj}^1(l) x_j^c(l) + \sum_{i=1}^{n} \omega_{ki}^2(l) u_i(l) \\ k = 1, 2, \ldots, N \end{gathered} \tag{6}$$

Now, $x_j^c(l)$ designates the signal that is transported from the k-th context layer, and $\omega_{kj}^1(l)$ describes the i-th and j-th weights of the HLs directed from the o-th nodes. Therefore, for the input layer I, the weight of HL k is obtained from $\omega_{ki}^2(l)$. This is accomplished using:

$$W_k(l) = f_0(\overline{v}_k(l)) \tag{7}$$

$$\overline{v}_k(l) = \frac{v_k(l)}{\max\{v_k(l)\}} \tag{8}$$

which indicates the standardized value of the HL. The next layer is the context layer. Here, the outcome is equivalent to the subsequent expression:

$$C_k(l) = \beta C, (l-1) + W_k(l-1),\ k = 1, 2, \ldots, N \tag{9}$$

In Equation (9), W_k indicates the self-connected feedback amongst [0, 1] and it is given as follows:

$$y_0(l) = \sum_{k=1}^{N} \omega_{ok}^3(l) W_k, (l),\ 0 = 1, 2, \ldots, n \tag{10}$$

In Equation (10), ω_{ok}^3 determines the weight connecting the k-th to the o-th layers. Figure 2 defines the framework of the ENN technique.

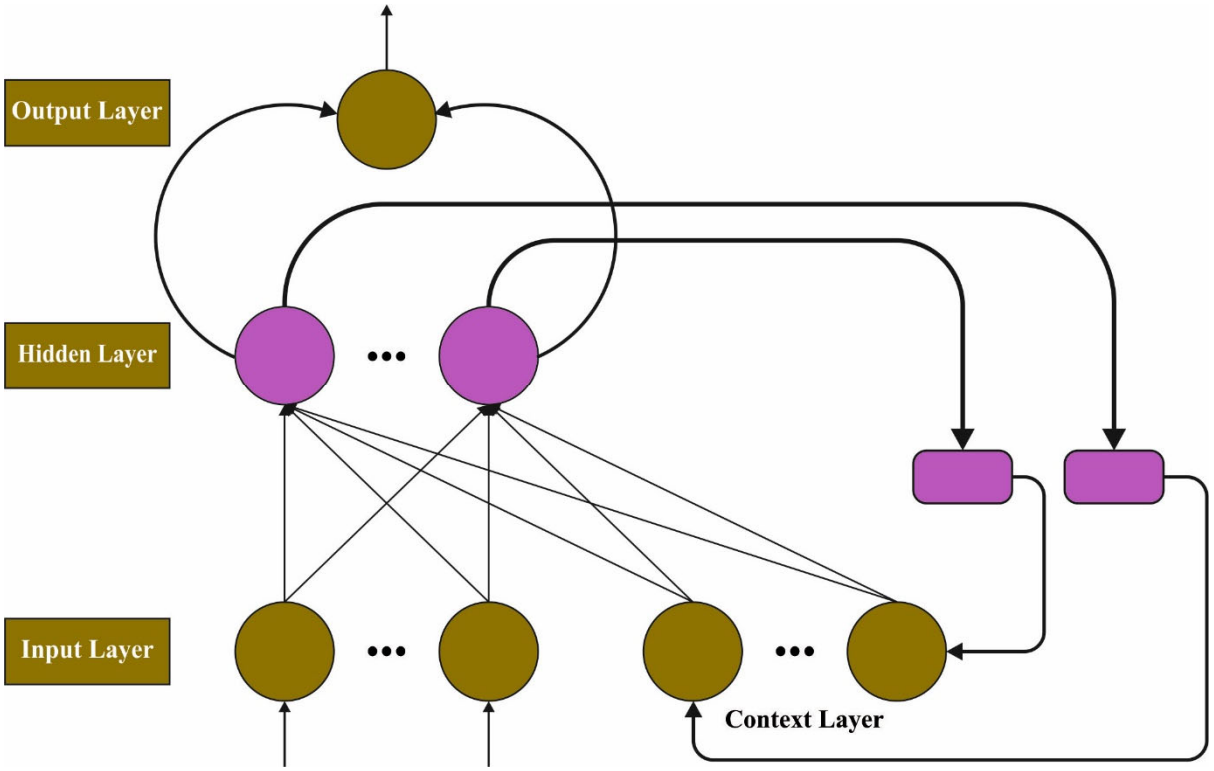

Figure 2. Structure of the ENN.

To optimally choose the ENN parameters, the CSO algorithm was applied in this work.

3.4. Parameter Tuning Using CSO Algorithm

Finally, the weight of the abovementioned ENN model was improved based on the CSO algorithm. The CSO algorithm is stimulated from the hierarchy and foraging behavior of chicken flocks [21]. In the CSO, the location of every individual was considered a candidate solution to the optimized problem. The count of individuals in the chicken flock can be represented as N. Each individual searches for food in a D dimensional space and upgrades their identity at each $\mathcal{G}$ generation. The sequence of serial numbers of each individual can be $\{1, 2, 3, \ldots,\ N_r, N_r + 1, \ldots,\ tN_hN_h + 1, \ldots,\ N_c\}$, whereas N_r, N_h, and N_c indicate the maximal serial numbers of the roosters, hens, and chicks in sub-flock afterward sorting, correspondingly. The original and updated location of each rooster is determined as $X_{i,j}$ and $X_{i,j}^{new}$, whereby $i \in \{1, 2, 3, \ldots, N_r\}$, $j \in \{1, 2, 3, \ldots,\ D\}$. Roosters with low fitness can forage food in a wide search area, as is shown in the following.

$$X_{i_G j}^{new} = X_{i,j} \cdot \left(1 + \Re and n\left(0, \sigma^2\right)\right) \tag{11}$$

$$\sigma^2 = \begin{cases} \exp\left(\frac{(f_n - f_i)}{|f_i| + \zeta}\right)_t \ if\ f_i > f_n \\ 1_t\ otherwise \end{cases} \tag{12}$$

Now, $\Re andn\left(0, \sigma^2\right)$ indicates the uniformly distributed random number with an average value of 0 and standard deviation of σ^2. f shows the fitness function. f_i and f_n denote the fitness value of the i-th rooster and n-th rooster, correspondingly, whereas $i, n \in \{1, 2, 3, \ldots, N_r\}$ and $i \neq n$. ζ denotes a number closer to 0, which is utilized to prevent the denominator $|f_i| + \zeta$ from being 0. The original and upgraded locations of hens are determined by $X_{i,j}$ and $X_{i,j}^{new}$, whereas $i \in \{N_r + 1, N_r + 2, \ldots, N_h\}, j \in \{1, 2, 3, \ldots, D\}$. $X_{c,j}$ denotes the location of the spouse, and $X_{d,j}$ indicates the location of the individual, where i hens want to steal food, in which $c \in \{1, 2, 3, \ldots, tN_r\}, d \in \{1, 2, 3, \ldots, N_h\}$. The searching and stealing capabilities of the hens are associated with their fitness value.

$$X_{i,j}^{new} = X_{i,j} + S_1 \cdot \Re and \cdot \left(X_{c,j} - X_{i,j}\right) + S_2 \cdot \Re and \cdot \left(X_{d,j} - X_{i,j}\right) \tag{13}$$

$$S_1 = \exp\left(\frac{f_i - f_C}{|f_i| + \zeta}\right) \tag{14}$$

$$S_2 = \exp\left(f_d - f_i\right) \tag{15}$$

Now, $\Re and$ indicates a random integer ranging from [0, 1]. The chick follows its mother for food foraging. The small fitness value makes it simple for them to search food by foraging as follows:

$$X_{i,j}^{new} = X_{i,j} + \varpi \cdot \left(X_{m,j} - X_{i,j}\right) \tag{16}$$

In Equation (16), $X_{i,j}$ and $X_{i,j}^{new}$ indicate the original and upgraded locations of chicks, correspondingly. For every chick, $i \in 224\,\{N_h + 1, N_h + 2, \ldots, N_m\}$, and $j \in \{1, 2, 3, \ldots, D\}$. $X_{m,j}$ shows the location of the mother hen analogous to the i-th chicks, where $m \in \{N_\gamma + 1, N_\gamma + 2, \ldots, N_h\}$. ϖ refers to the succeeding probability for all the chicks that follow their mother hen to forage. Considering the variances among all the chicks, ϖ is generated at random among [0, 2] as presented in Algorithm 1.

Algorithm 1. Pseudo-Code of the Original CSO Algorithm

Input: Randomly allocated primary values to individuals from the chicken flock; Determine parameters namely N_r, N_h, and N_c;
$t = 0$;
Compute fitness values to all individuals
While $t < T_{\max}$ do
If $(t \,\%\, \mathcal{G} == 0)$
    Rank the fitness values and classifies the flock as distinct sub-groups;
End if
For $i = 1 : N_r$
    Upgrade $X_{i,j}$ with Equation (11);
End for
For $i = (N_r + 1) : N_m$
    Upgrade $X_{i,j}$ with Equation (13);
End for
For $i = (N_m + 1) : N_c$
    Update $X_{i,j}$ with Equation (16);
End for
If $f\left(X_{i,j}\right) < f(X_{i,j}^{new})$
    $X_{i,j} = X_{i,j}$;
Else
    $X_{i,j} = X_{i,j}^{new}$;
End if
    $t = t + 1$;
End while
Output: Optimal hyperparameter values

The CSO algorithm derives a fitness function (FF) to obtain an improved classifier outcome. In this study, the reduced classifier error rate is considered as the FF, as given below in Equation (17).

$$\begin{aligned} fitness(x_i) &= ClassifierErrorRate(x_i) \\ &= \frac{number\ of\ misclassified\ samples}{Total\ number\ of\ samples} * 100 \end{aligned} \tag{17}$$

4. Results and Discussion

The proposed model was simulated using Python 3.6.5. The experiments for the proposed model used a PC i5-8600k, GeForce 1050Ti 4 GB, 16 GB RAM, 250 GB SSD, and 1 TB HDD. The parameter settings were as follows: learning rate: 0.01, dropout: 0.5, batch size: 5, epoch count: 50, and activation: ReLU.

This section inspects the bot net classification results of the BND-BMOML model on the N_BaIoT [22] dataset. The dataset comprises 17,001 samples with three class labels. Table 1 provides a detailed explanation of the dataset.

Table 1. Dataset details.

Class	No. of Samples
Benign	5000
Mirai	7001
Gafgyt	5000
Total Number of Samples	**17,001**

Figure 3 illustrates the confusion matrices formed by the BND-BMOML model. In the entire dataset, the BND-BMOML model categorized 4933 samples into the benign class, 6963 samples into the Mirai class, and 4925 samples into Gafgyt. Moreover, in 70% of the TR dataset, the BND-BMOML method categorized 3425 samples into the benign class, 4900 samples into the Mirai class, and 3447 samples into Gafgyt. Next, on 30% of the TS dataset, the BND-BMOML approach categorized 1508 samples into the benign class, 2063 samples into the Mirai class, and 1478 samples into Gafgyt.

Table 2 and Figure 4 offer brief botnet detection results using the BND-BMOML model for the entire dataset. The results show that the BND-BMOML model achieved enhanced results under all classes. For instance, in the benign class, the BND-BMOML model provided an $accu_y$ of 99.18%, $prec_n$ of 98.54%, $reca_l$ of 98.66%, F_{score} of 98.60%, and MCC of 98.02%. Furthermore, in the Mirai class, the BND-BMOML technique offered $accu_y$ of 99.52%, $prec_n$ of 99.39%, $reca_l$ of 99.46%, F_{score} of 99.42%, and MCC of 99.02%. Finally, in the Gafgyt class, the BND-BMOML approach presented $accu_y$ of 99.18%, $prec_n$ of 98.72%, $reca_l$ of 98.50%, F_{score} of 98.61%, and MCC of 98.03%.

Table 2. Result analysis for BND-BMOML algorithm with distinct class labels using entire dataset.

	Entire Dataset				
Labels	**Accuracy**	**Precision**	**Recall**	**F-Score**	**MCC**
Benign	99.18	98.54	98.66	98.60	98.02
Mirai	99.52	99.39	99.46	99.42	99.02
Gafgyt	99.18	98.72	98.50	98.61	98.03
Average	**99.29**	**98.88**	**98.87**	**98.88**	**98.35**

Table 3 and Figure 5 portray the detailed botnet detection results for the BND-BMOML methodology with 70% of the TR dataset. The BND-BMOML approach demonstrated enhanced results with all classes. For example, in the benign class, the BND-BMOML model offered $accu_y$ of 99.12%, $prec_n$ of 98.45%, $reca_l$ of 98.53%, F_{score} of 98.49%, and MCC of 97.87%. Additionally, in the Mirai class, the BND-BMOML algorithm rendered $accu_y$ of

99.54%, $prec_n$ of 99.39%, $reca_l$ of 99.49%, F_{score} of 99.44%, and MCC of 99.05%. Moreover, in the Gafgyt class, the BND-BMOML method achieved $accu_y$ of 99.19%, $prec_n$ of 98.74%, $reca_l$ of 98.51%, F_{score} of 98.63%, and MCC of 98.06%.

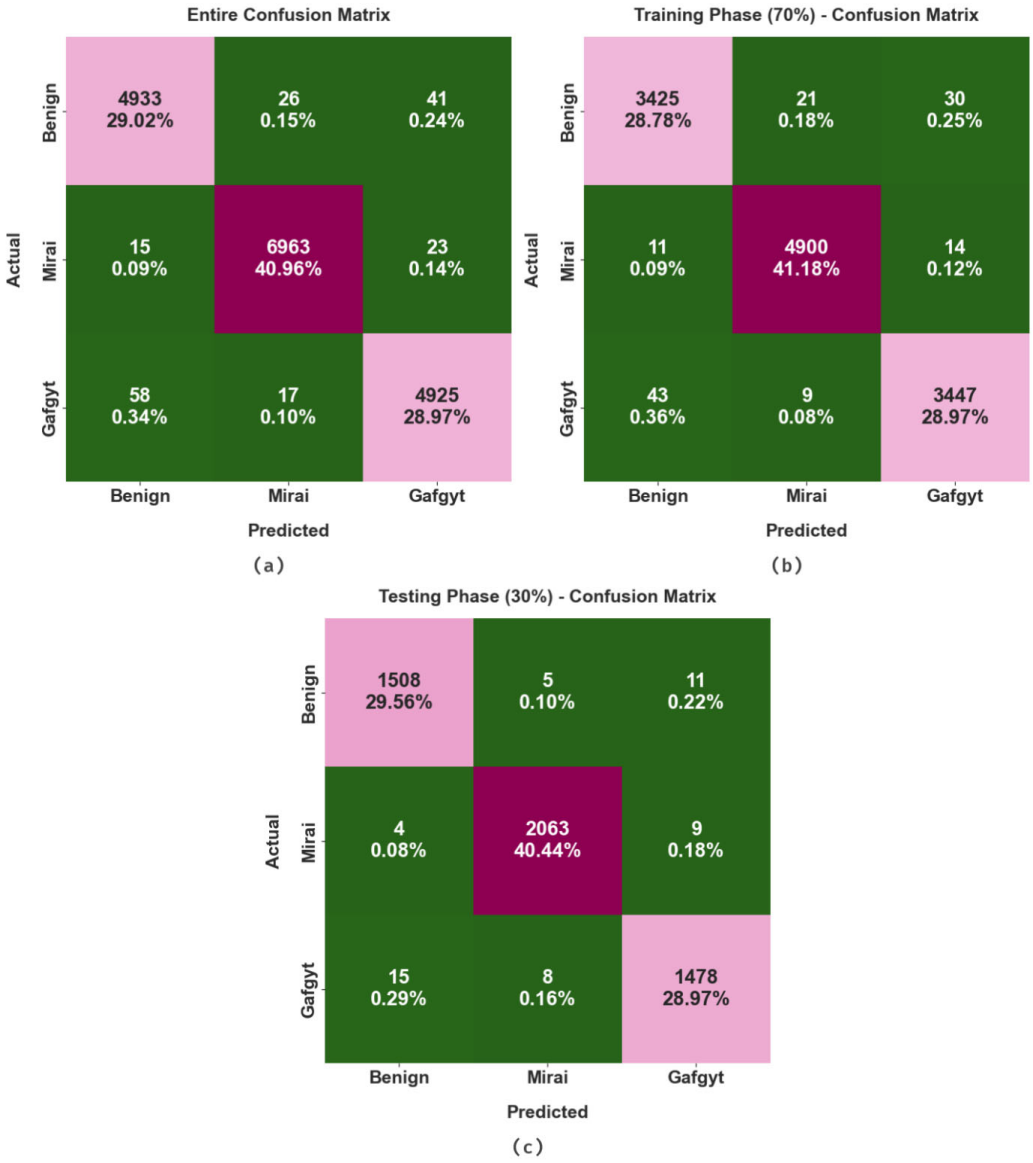

Figure 3. Confusion matrices for BND-BMOML approach: (**a**) entire dataset, (**b**) 70% of TR data, and (**c**) 30% of TS data.

Table 3. Result analysis for BND-BMOML algorithm with distinct class labels using 70% of the TR data.

Training Phase (70%)					
Labels	**Accuracy**	**Precision**	**Recall**	**F-Score**	**MCC**
Benign	99.12	98.45	98.53	98.49	97.87
Mirai	99.54	99.39	99.49	99.44	99.05
Gafgyt	99.19	98.74	98.51	98.63	98.06
Average	**99.28**	**98.86**	**98.85**	**98.85**	**98.32**

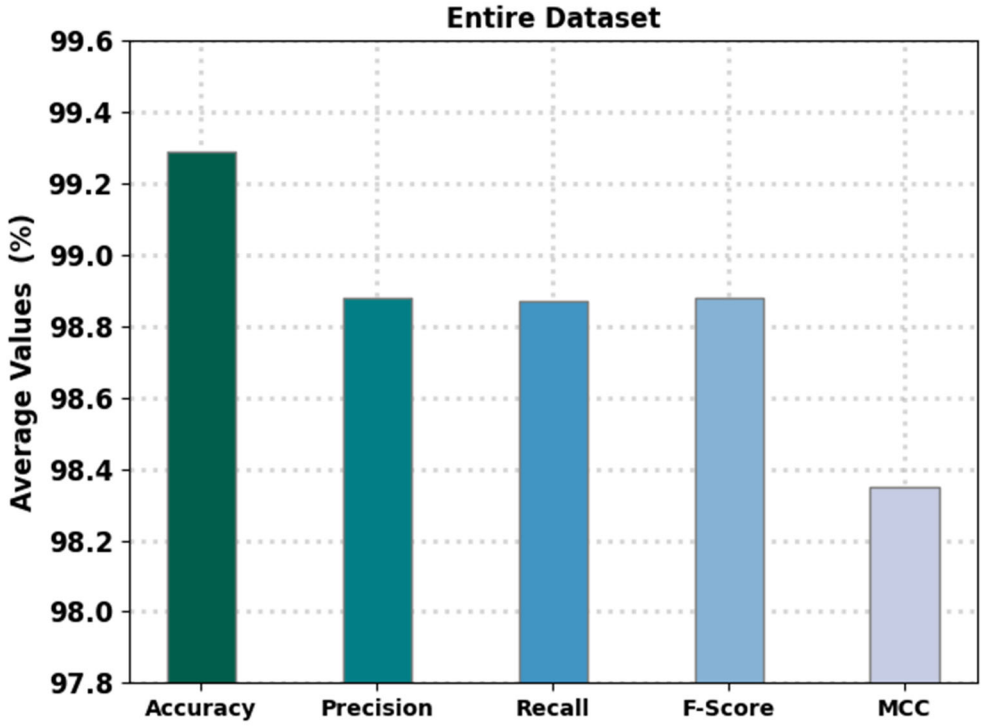

Figure 4. Average analysis of BND-BMOML algorithm using entire dataset.

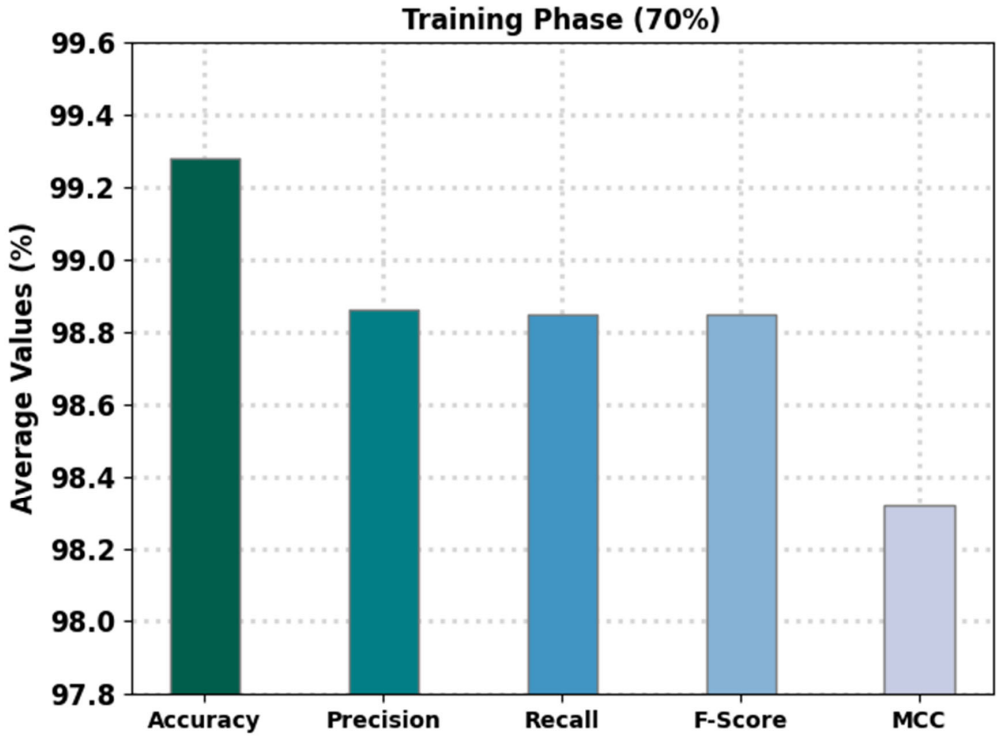

Figure 5. Average analysis of BND-BMOML algorithm using 70% of the TR data.

Table 4 and Figure 6 present brief botnet detection results for the BND-BMOML technique using 30% of the TS dataset. The BND-BMOML method displayed enhanced results in every class label. For example, in the benign class, the BND-BMOML algorithm rendered $accu_y$ of 99.31%, $prec_n$ of 98.76%, $reca_l$ of 98.95%, F_{score} of 98.85%, and MCC of 98.36%. Additionally, in the Mirai class, the BND-BMOML technique presented $accu_y$ of

99.49%, $prec_n$ of 99.37%, $reca_l$ of 99.37%, F_{score} of 99.37%, and MCC of 98.94%. Furthermore, in the Gafgyt class, the BND-BMOML method, offered $accu_y$ of 99.16%, $prec_n$ of 98.66%, $reca_l$ of 98.47%, F_{score} of 98.57%, and MCC of 97.97%.

Table 4. Result analysis for BND-BMOML algorithm with distinct class labels using 30% of the TS data.

Testing Phase (30%)					
Labels	**Accuracy**	**Precision**	**Recall**	**F-Score**	**MCC**
Benign	99.31	98.76	98.95	98.85	98.36
Mirai	99.49	99.37	99.37	99.37	98.94
Gafgyt	99.16	98.66	98.47	98.57	97.97
Average	**99.32**	**98.93**	**98.93**	**98.93**	**98.43**

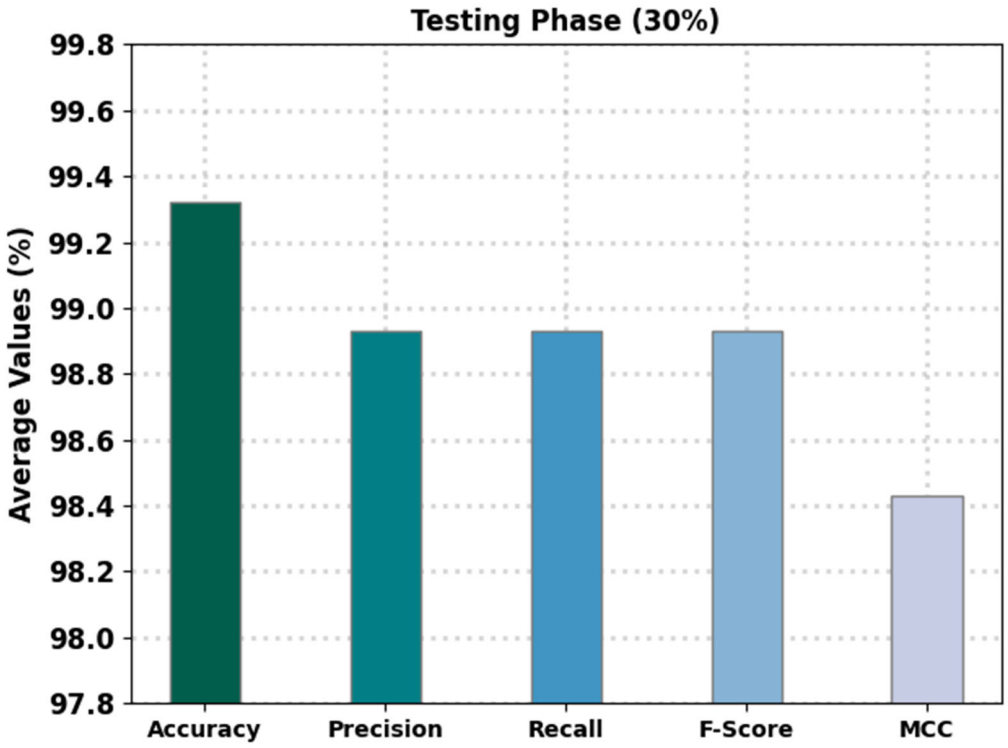

Figure 6. Average analysis for BND-BMOML algorithm using 30% of the TS data.

The training accuracy (TRA) and validation accuracy (VLA) attained by the BND-BMOML method in the test dataset are shown in Figure 7. The experimental outcome shows that the BND-BMOML algorithm gained maximum values for TRA and VLA. The VLA was seemingly greater than the TRA.

The training loss (TRL) and validation loss (VLL) acquired by the BND-BMOML technique in the test dataset are displayed in Figure 8. The experimental outcome shows that the BND-BMOML technique exhibited minimal values for the TRL and VLL. Particularly, the VLL was less than the TRL.

A clear precision-recall analysis of the BND-BMOML algorithm in the test dataset is exemplified in Figure 9. The figure shows that the BND-BMOML algorithm resulted in enhanced precision-recall values in every class label.

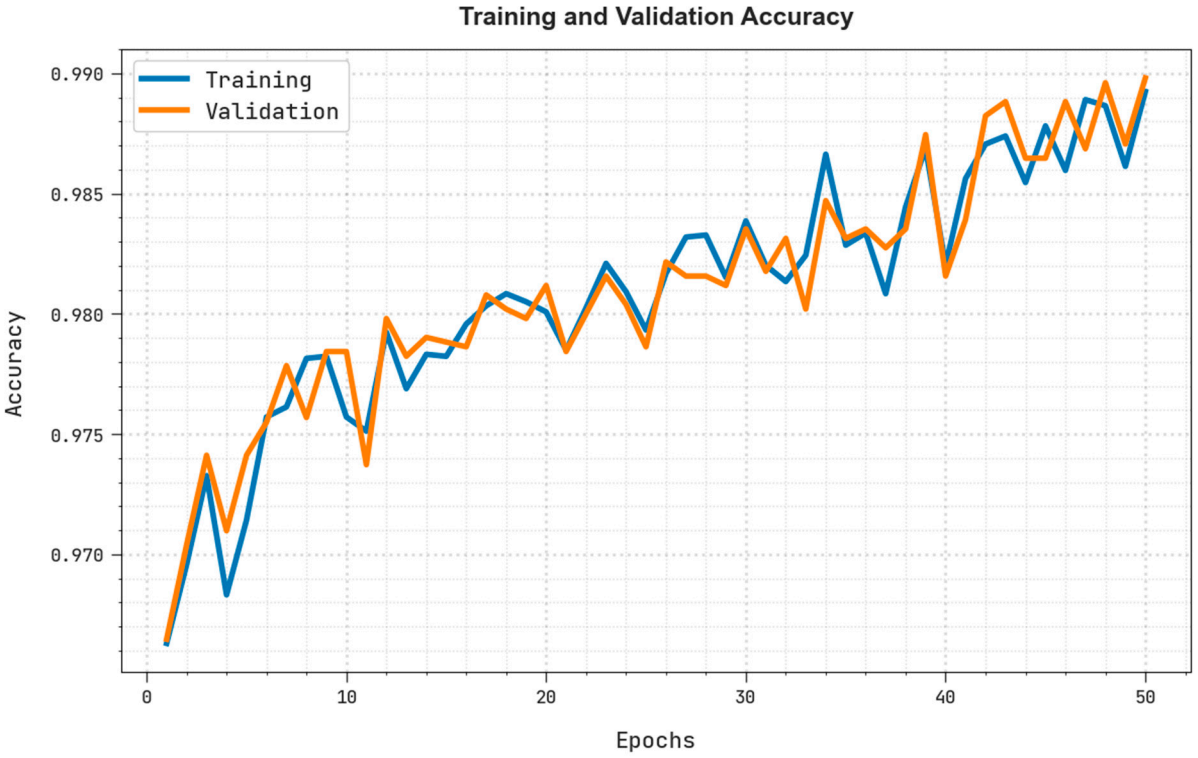

Figure 7. TRA and VLA analysis for BND-BMOML algorithm.

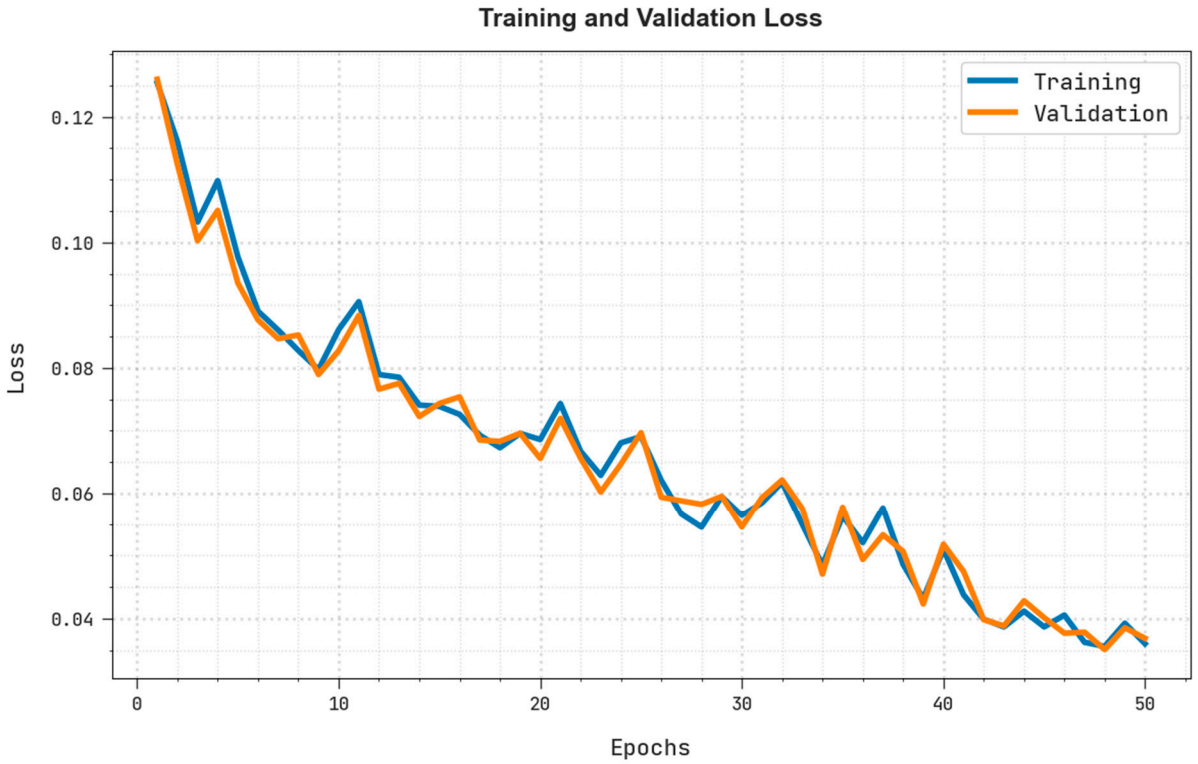

Figure 8. TRL and VLL analysis for BND-BMOML algorithm.

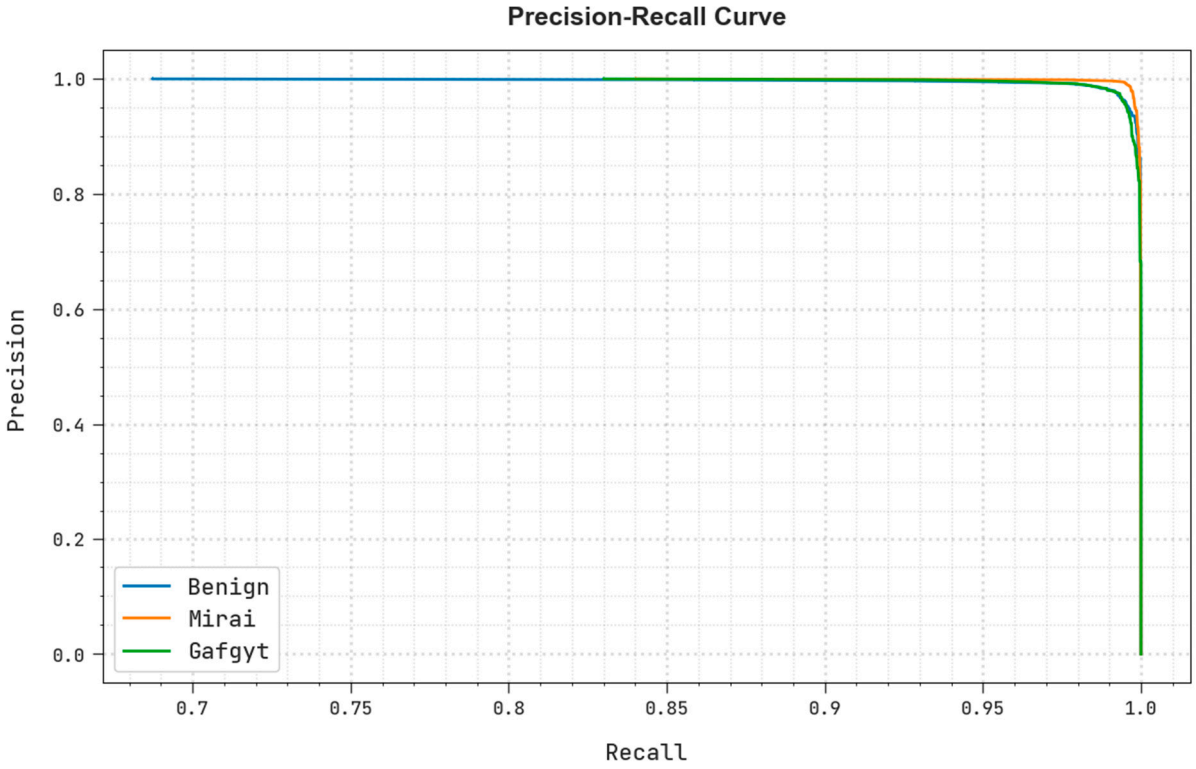

Figure 9. Precision-recall curve analysis for BND-BMOML algorithm.

Table 5 and Figure 10 offer a detailed comparative study of the BND-BMOML model and existing models [23]. The results indicate that the DBN, LSTM, and CNN-RNN models reported lower classification performance. Next, the LSTM-CNN and DNN models achieved slightly higher classifier results. Though the DNN-LSTM model reached reasonable performance with a classification accuracy of 99.11, the BND-BMOML model shows the maximum accuracy of 99.32%.

Table 5. Comparative analysis of BND-BMOML approach and recent algorithms.

Methods	Accuracy	Precision	Recall	F-Score
BND-BMOML	99.32	98.93	98.93	98.93
DNN-LSTM	99.11	98.37	98.24	98.16
LSTM	97.14	95.93	94.62	95.18
CNN-RNN	96.41	94.01	97.58	94.03
LSTM-CNN	98.85	96.98	97.62	96.15
DNN	98.82	96.98	96.37	94.75
DBN	96.92	95.40	96.85	96.07

Finally, a brief running time (RUNT) examination of the BND-BMOML model and recent models is provided in Table 6 and Figure 11. The attained results show that the LSTM and DBN models reported higher RUNTs of 3.86 ms and 3.97 ms. Along with that, the DNN-LSTM and CNN-RNN models attained slightly improved RUNTs of 1.27 ms and 1.15 ms, respectively. The LSTM-CNN and DNN models revealed reasonable RUNTs of 0.35 ms and 0.59 ms, respectively. However, the BND-BMOML model showed superior results, with a minimal RUNT of 0.17 ms. Thus, the BND-BMOML model was found to be better than existing models.

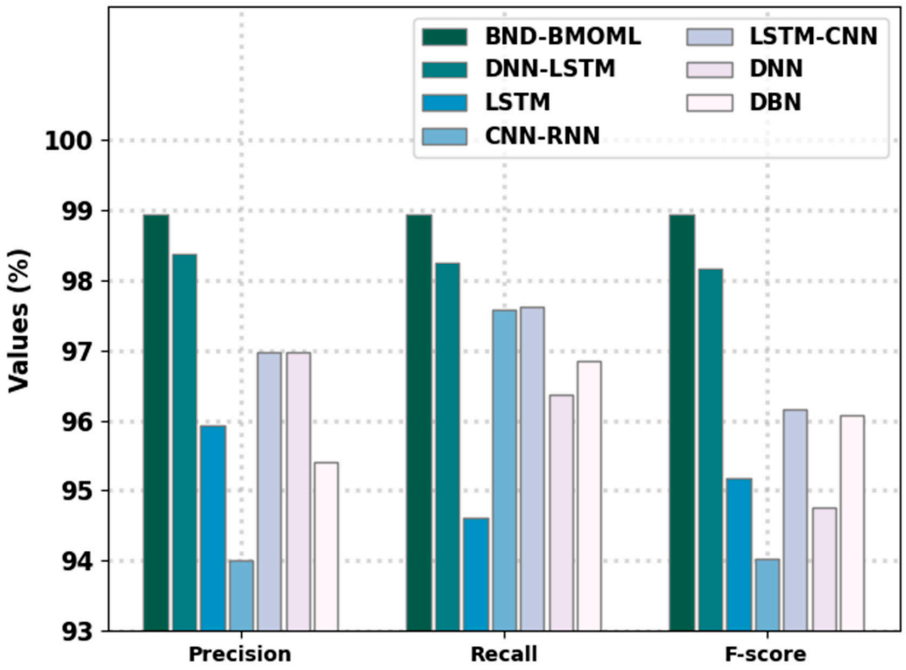

Figure 10. Comparative analysis of BND-BMOML approach and recent algorithms.

Table 6. Running time analysis for BND-BMOML approach and recent algorithms.

Methods	Running Time (ms)
BND-BMOML	0.17
DNN-LSTM	1.27
LSTM	3.86
CNN-RNN	1.15
LSTM-CNN	0.35
DNN	0.59
DBN	3.97

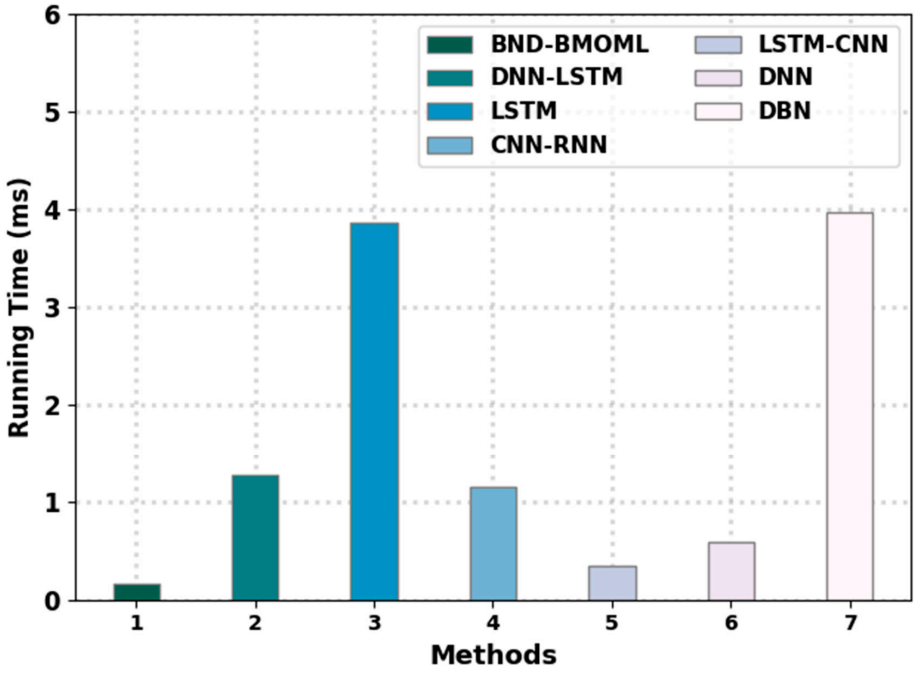

Figure 11. Running time analysis of BND-BMOML approach and recent algorithms.

5. Conclusions

In this article, a new BND-BMOML system was established for the identification and recognition of botnets in the IoT environment. To accomplish this, the BND-BMOML model initially follows the data standardization approach. In the presented BND-BMOML model, the BMO algorithm is employed to select a useful set of features. For botnet detection, the BND-BMOML technique employs the ENN model in this study. Finally, the presented BND-BMOML model uses the CSO algorithm for parameter tuning process. Experimental validation of the BND-BMOML approach was applied using a benchmark dataset and the results portrayed a significant improvement in performance over existing methods. Thus, the presented BND-BMOML technique can be exploited for the real-time botnet detection process. In the future, the performance of the BND-BMOML technique can be improved using advanced DL models.

Author Contributions: Conceptualization, F.S.A. and M.M.; methodology, A.G.; software, A.M.; validation, A.Y., R.A. and O.A.; formal analysis, H.M.; investigation, A.Y.; resources, H.M.; data curation, A.M.; writing—original draft preparation, F.S.A., M.M., A.Y. and A.G.; writing—review and editing, R.A., O.A. and H.M.; visualization, A.M.; supervision, F.S.A.; project administration, A.M.; funding acquisition, F.S.A. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This article does not contain any studies with human participants performed by any of the authors.

Informed Consent Statement: Not applicable.

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

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through the Small Groups Project under grant number (168/43) and Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R319), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work with Grant Code: 22UQU4320484DSR02.

Conflicts of Interest: The authors declare that they have no conflict of interest. The manuscript was written through the contributions of all authors. All authors have given approval to the final version of the manuscript.

References

1. Pokhrel, S.; Abbas, R.; Aryal, B. IoT security: Botnet detection in IoT using machine learning. *arXiv* **2021**, arXiv:2104.02231.
2. Chen, Z. Research on Internet Security Situation Awareness Prediction Technology based on Improved RBF Neural Network Algorithm. *J. Comput. Cogn. Eng.* **2022**, *1*, 103–108.
3. Shinan, K.; Alsubhi, K.; Alzahrani, A.; Ashraf, M.U. Machine learning-based botnet detection in software-defined network: A systematic review. *Symmetry* **2021**, *13*, 866. [CrossRef]
4. Vasko, F.J.; Lu, Y.; McNally, B. A Simple Methodology that Efficiently Generates All Optimal Spanning Trees for the Cable-Trench Problem. *J. Comput. Cogn. Eng.* **2022**, *1*, 13–20.
5. Namasudra, S.; Crespo, R.G.; Kumar, S. Introduction to the special section on advances of machine learning in cybersecurity (VSI-mlsec). *Comput. Electr. Eng.* **2022**, *100*, 108048. [CrossRef]
6. Gutub, A. Boosting image watermarking authenticity spreading secrecy from counting-based secret-sharing. *CAAI Trans. Intell. Technol.* **2022**, *2*, 1–13. [CrossRef]
7. Das, S.; Namasudra, S. Multi-Authority CP-ABE-Based Access Control Model for IoT-Enabled Healthcare Infrastructure. *IEEE Trans. Ind. Inform.* 2022; *in press*. [CrossRef]
8. Alauthman, M.; Aslam, N.; Al-Kasassbeh, M.; Khan, S.; Al-Qerem, A.; Choo, K.K.R. An efficient reinforcement learning-based Botnet detection approach. *J. Netw. Comput. Appl.* **2020**, *150*, 102479. [CrossRef]
9. Sarkar, S.; Saha, K.; Namasudra, S.; Roy, P. An efficient and time saving web service based android application. *SSRG Int. J. Comput. Sci. Eng.* **2015**, *2*, 18–21.
10. Wani, A.; Khaliq, R. SDN-based intrusion detection system for IoT using deep learning classifier (IDSIoT-SDL). *CAAI Trans. Intell. Technol.* **2021**, *6*, 281–290. [CrossRef]

11. Vinayakumar, R.; Alazab, M.; Srinivasan, S.; Pham, Q.V.; Padannayil, S.K.; Simran, K. A visualized botnet detection system based deep learning for the internet of things networks of smart cities. *IEEE Trans. Ind. Appl.* **2020**, *56*, 4436–4456. [CrossRef]
12. Al Shorman, A.; Faris, H.; Aljarah, I. Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT botnet detection. *J. Ambient Intell. Human. Comput.* **2020**, *11*, 2809–2825. [CrossRef]
13. Popoola, S.I.; Adebisi, B.; Ande, R.; Hammoudeh, M.; Anoh, K.; Atayero, A.A. smote-drnn: A deep learning algorithm for botnet detection in the internet-of-things networks. *Sensors* **2021**, *21*, 2985. [CrossRef] [PubMed]
14. Sriram, S.; Vinayakumar, R.; Alazab, M.; Soman, K.P. Network flow based IoT botnet attack detection using deep learning. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6 July 2020; pp. 189–194.
15. Habib, M.; Aljarah, I.; Faris, H.; Mirjalili, S. Multi-objective particle swarm optimization for botnet detection in internet of things. In *Evolutionary Machine Learning Techniques*; Springer: Singapore, 2020; pp. 203–229.
16. Wu, D.; Fang, B.; Wang, J.; Liu, Q.; Cui, X. Evading machine learning botnet detection models via deep reinforcement learning. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
17. Alrashdi, I.; Alqazzaz, A.; Aloufi, E.; Alharthi, R.; Zohdy, M.; Ming, H. Ad-IoT: Anomaly detection of IoT cyberattacks in smart city using machine learning. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 305–310.
18. McDermott, C.D.; Majdani, F.; Petrovski, A.V. Botnet detection in the internet of things using deep learning approaches. In Proceedings of the 2018 IEEE International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
19. Houssein, E.H.; Abdelminaam, D.S.; Hassan, H.N.; Al-Sayed, M.M.; Nabil, E. A hybrid barnacles mating optimizer algorithm with support vector machines for gene selection of microarray cancer classification. *IEEE Access* **2021**, *9*, 64895–64905. [CrossRef]
20. Ding, L.; Bai, Y.; Liu, M.D.; Fan, M.H.; Yang, J. Predicting short wind speed with a hybrid model based on a piecewise error correction method and Elman neural network. *Energy* **2022**, *244*, 122630. [CrossRef]
21. Liang, S.; Fang, Z.; Sun, G.; Liu, Y.; Qu, G.; Zhang, Y. Sidelobe reductions of antenna arrays via an improved chicken swarm optimization approach. *IEEE Access* **2020**, *8*, 37664–37683. [CrossRef]
22. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
23. Sattari, F.; Farooqi, A.H.; Qadir, Z.; Raza, B.; Nazari, H.; Almutiry, M. A Hybrid Deep Learning Approach for Bottleneck Detection in IoT. *IEEE Access* **2022**, *10*, 77039–77053. [CrossRef]