electronics

MDPI

Article

Botnet Attack Detection Using Local Global Best Bat Algorithm for Industrial Internet of Things

Abdullah Alharbi [1], Wael Alosaimi [1], Hashem Alyami [2], Hafiz Tayyab Rauf [3,*] and Robertas Damaševičius [4]

[1] Department of Information Technology, College of Computers and Information Technology, Taif University, P. O. Box 11099, Taif 21944, Saudi Arabia; amharbi@tu.edu.sa (A.A.); w.osaimi@tu.edu.sa (W.A.)
[2] Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, Taif 21944, Saudi Arabia; hyami@tu.edu.sa
[3] Centre for Smart Systems, AI and Cybersecurity, Staffordshire University, Stoke-on-Trent ST4 2DE, UK
[4] Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland; robertas.damasevicius@polsl.pl
* Correspondence: hafiztayyabrauf093@gmail.com

Abstract: The need for timely identification of Distributed Denial-of-Service (DDoS) attacks in the Internet of Things (IoT) has become critical in minimizing security risks as the number of IoT devices deployed rapidly grows globally and the volume of such attacks rises to unprecedented levels. Instant detection facilitates network security by speeding up warning and disconnection from the network of infected IoT devices, thereby preventing the botnet from propagating and thereby stopping additional attacks. Several methods have been developed for detecting botnet attacks, such as Swarm Intelligence (SI) and Evolutionary Computing (EC)-based algorithms. In this study, we propose a Local-Global best Bat Algorithm for Neural Networks (LGBA-NN) to select both feature subsets and hyperparameters for efficient detection of botnet attacks, inferred from 9 commercial IoT devices infected by two botnets: Gafgyt and Mirai. The proposed Bat Algorithm (BA) adopted the local-global best-based inertia weight to update the bat's velocity in the swarm. To tackle with swarm diversity of BA, we proposed Gaussian distribution used in the population initialization. Furthermore, the local search mechanism was followed by the Gaussian density function and local-global best function to achieve better exploration during each generation. Enhanced BA was further employed for neural network hyperparameter tuning and weight optimization to classify ten different botnet attacks with an additional one benign target class. The proposed LGBA-NN algorithm was tested on an N-BaIoT data set with extensive real traffic data with benign and malicious target classes. The performance of LGBA-NN was compared with several recent advanced approaches such as weight optimization using Particle Swarm Optimization (PSO-NN) and BA-NN. The experimental results revealed the superiority of LGBA-NN with 90% accuracy over other variants, i.e., BA-NN (85.5% accuracy) and PSO-NN (85.2% accuracy) in multi-class botnet attack detection.

Keywords: botnet attacks; intrusion detection; heuristic optimization; neural networks; bat algorithm; Internet-of-Things security

Citation: Alharbi, A.; Alosaimi, W.; Alyami, H.; Rauf, H.T.; Damaševičius, R. Botnet Attack Detection Using Local Global Best Bat Algorithm for Industrial Internet of Things. *Electronics* **2021**, *10*, 1341. https://doi.org/10.3390/electronics10111341

Academic Editor: Constantinos Kolias

Received: 9 March 2021
Accepted: 29 March 2021
Published: 3 June 2021

1. Introduction

An increase in cyber-crimes has made the detection of intrusions in the network a vital research area [1]. Traditionally, personal computers (PC) and computer networks have been the subject of cyber-attack, but recently, cyber-physical systems [2], Internet of Things (IoT) [3], internet of medical things (IoMT) [4], Internet of Connected Vehicles [5], smart factories [6], and 5G communication infrastructures [7] have become targets of numerous attacks as well. Several studies have been carried out to suggest remarkable strategies to fight against cyber actions [8,9]. Previous strategies are not able to resolve complex cyber-attacks. Common preventive approaches such as authentication, firewalls, and antivirus are not sufficient for complicated cyber-attacks [10]. Algorithms used for classification

can be utilized to make decisions while detecting intrusions in networks [11]. Various machine learning algorithms such as decision trees, K nearest neighbor (KNN), support vector machine (SVM), Naive Bayes, neural networks, random forest, and fuzzy logic are being used for intrusion detection [12–14].

The performance of each algorithm can be improved in the case of a group or combination strategy. Group model is a renowned strategy of machine learning in which several algorithms are utilized to enhance predictions and speed of detection. The suggested method focused on the use of supervised algorithms of machine learning to suggest a framework for classification through stacked generalization. Big data is being produced by IoT devices and networks in the current era [15], causing difficulties to secure the industry and academia. Consequently, various threats and malware are rising and cannot be handled by using current strategies [16].

Botnet attacks such as Gafgyt [17] and Mirai [18] are on the rise. A botnet is an computer network consisting of several hosts running standalone software. A bot in such a network is the computer itself with malware that allows an attacker to perform certain actions using the resources of an infected PC.

A botnet runs a bot on several machines linked to the internet to form a botnet operated by a malicious group [19]. Botnets pose a significant threat to network security as they are commonly used for many Internet crimes such as Distributed Denial-of-Service (DDoS) attacks, identity theft, email spam, and click fraud. Botnet-based DDoS attacks are devastating for the target network as they will drain all network bandwidth and victim computer resources and cause an interruption of services. Machine learning methods are now commonly used to track these attacks in IoT [20–22].

The botnet can be instructed in targeted, distributed denial of service attacks on any system on the Internet to consume resources (for example, bandwidth ability) of the system in such a way that it cannot properly serve their legitimate users. Nowadays, practically all DDoS attacks are carried out from a botnet platform. Despite its simplicity, the DDoS attack technique is very effective due to the size of the botnet and the total bandwidth of the bots.

An intruder can illegally access a user's information or slow down the system by launching several attacks. A computer network can be targeted by several attacks such as User to Root (U2R), Probing (Probe), Remote to Local (R2L), Denial of Service (DoS), Port Scanning, Brute Force, etc. Any transport, application, or protocol such as Internet Control Message Protocol (ICMP), File transfer protocol (FTP), User Datagram Protocol (UDP), Simple Mail Transfer Protocol (SMTP), Transmission Control Protocol (TCP), Hypertext Transfer Protocol (HTTP), etc., can be used to execute these attacks. Network-based Intrusion Detection Systems (NIDS) are suggested to be used to handle such attacks to scan the network and to identify such attacks [23,24].

Several machine learning methods have been used to design intrusion detection systems, but these methods fail to handle large traffic. Similarly, deep learning methods lack the optimization perspective that leads to low generality of the model. Researchers are now able to develop high-detection detectors for fixed botnet data sets with binary recognition tasks [25]. However, since botnet attack detection techniques are constantly changing, a high detection rate for only fixed data sets cannot guarantee excellent accuracy when dealing with complex traffic data. Thus, there is a need for neuro-evolution-based classification techniques that can help decide the number of layers and neurons during the detection process [26].

The hyperparameters should be optimized during neural network training and weight learning processes. Many fixed optimizers have been proposed in modern literature for this reason, such as rmsprop, Adam, adagrad, adamax, and adadelta [27–29]. However, they miss the problem nature and data set dimension; therefore, several evolutionary methods [30] have been employed to overcome this issue and to optimize the hyperparameters.

In this research, we propose a neural network-based optimized method Local-Global best Bat Algorithm for Neural Network (LGBA-NN) to determine both feature subsets

and hyperparameters for efficient detection of botnet attacks. LGBA-NN further contains three major improvements; the first modification involves robust population initialization of bats by Gaussian distribution. The proposed population initialization supports the bat's solution in producing robust offspring by providing sufficient diversity. Secondly, the proposed BA uses a local-global best-based inertia weight to update the velocity of the entire swarm bat. Finally, a local search mechanism based on the Gaussian density function and the local-global best function is proposed to improve exploration during each phase.

The novelty and contribution of this paper is summarized as follows:

- A novel meta-heuristic local-global best bat algorithm (LGBA) for optimization of the hyper-parameters of a neural network (LGBA-NN) is presented.
- The proposed LGBA-NN is tested on an N-BaIoT data set with extensive real-world traffic data with benign and malicious classes, achieving high performance.

2. Related Work

Anomaly-based detection and signature-based detection are the two methods of detection. For a known pattern of attack, signature-based detection is used. At the same time, anomaly-based detection is used for unknown and known patterns of attack [31]. Additionally, Network-based Intrusion Detection Systems (NIDS) depend on traffic identification, which means the flow of traffic is used to extract essential features used to classify traffic records into malicious or normal activities by utilizing machine learning algorithms [32]. Host-based and network-based systems are two classifications of intrusion detection systems. All actions are monitored in NIDS for detection of intrusion, which helps in identifying attacks such as Denial of Service (DoS) [33]. In contrast, in the case of Host-based Intrusion Detection Systems (HIDSs), specific approaches are implemented at significant points of the server or every system. A low rate of false alarms and highly accurate detection can be gained by utilizing a vast database in anomaly-based IDSs. For developing a particular database for anomaly-based systems, testing and training phases are implemented. In the training phase, regular patterns are specified, used for comparison during the testing phase.

Traditionally used signature and heuristic methods for detecting malicious software are not able to provide a sufficient level of detection of new and previously unknown variants of botnets. This determines the applicability of the machine learning methods in solving this problem. Advanced machine learning and deep learning methods are being utilized for security purposes while increasing the robustness and accuracy of attack detection without demanding advanced knowledge of security [34,35]. The efficiency of IDSs can be enhanced by utilizing nature-inspired meta-heuristic, grammar-based [36], data mining, machine learning, reinforcement learning, and artificial intelligence-based techniques [37–39]. The performance of IDS can be improved by utilizing techniques such as Artificial Bee Colony (ABC) [40], Particle Swarm Optimization (PSO) [41], Grey Wolf optimization [42], and Artificial Fish Swarm Algorithm [43]. Description of recent related works on network intrusion detection using several optimization, deep, and machine learning algorithms are given in Table 1.

One study [44] proposed a group model by utilizing a meta classification strategy facilitated through stacked generalization. UGR'16 and UNSW NB-15 are two varied data sets that were gathered from real and emulated network traffic. The proposed strategy showed an accuracy of 97%, while emulated data sets came up with an accuracy of 94%. Another study [45] proposed an algorithm based on double Particle Swarm Optimization (PSO) to choose hyperparameters and feature subsets in a single process. They utilized deep Belief Networks (DBN), Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN), and Deep Neural Networks (DNN) models for the proposed algorithm.

Tripathi et al. [46] utilized grasshopper optimization algorithm (GOA). Their suggested method was IDS-based to discriminate between malicious and regular traffic. Multilayer perceptron, naïve Bayes, decision tree, and support vector machine determine the

attack type. The CIC-IDS 2017 and KDD Cup 99 data sets were used to test their proposed method.

Table 1. Brief description of recent related works on network intrusion detection using several optimization, deep, and machine learning algorithms.

Ref	Data-Set	Methods	Evaluation Metrics	Accuracy
[44]	UNSW NB-15	Ensemble approach using concept of stacking	Accuracy, recall, and precision	Gained 97% accuracy
[45]	Two common IDS data sets	Double Particle Swarm Optimization (PSO)-based algorithm	Accuracy, recall, precision, specificity	Increased DR by 4% to 6%
[46]	KDD Cup 99 and CIC-IDS 2017	GOIDS	AUC, specificity, false-positive rate	Low false-positive rate.
[47]	Own collected	Immune Genetic Algorithm	Average standard deviation	85% of prediction accuracy
[48]	AWID data	M-AdaBoost-A	TPR/FPR, macro precision	Showed highly accurate results
[49]	NSL-KDD	SRDLM	Accuracy, precision, recall, F1	Attack detection over 99%
[50]	CICIDS 2017	Multi-stage optimized ML-based NIDS	Accuracy, precision, recall /TPR, FAR/FPR	Detection accuracy over 99%
[51]	UNSW-NB15	Feature selection model for NIDSs	TPR, TNR, FPR, FNR, accuracy, precision	Showed highly accurate results
[52]	CSE-CIC-IDS2018	ML and DL-based NIDS.	Accuracy, precision, recall, FAR, TNR	Gained promising results
[53]	CICIDS2017, NSL-KDD	LNNLS-KH	FPR, DR	Ensured high detection accuracy
[54]	KDD CUP 99	Data mining-based IDS	DR, FPR	Showed improved accuracy
[40]	NSL-KDD and ISCXIDS2012	Anomaly network-based IDS (A-NIDS)	TP, FP, FN, TN	Gained enhanced detection accuracy
[55]	KDDTest +	Improved convolutional neural network (ICNN)	AC, TPR, FPR	Higher recall rate and lower FPR
[56]	NSL-KDD	Network intrusion detection system (NIDS)	FPR, FNR, TPR, TNR	Low false alarm rates (FARs)
[57]	ISCX 2012, and Kyoto 2006+.	Hybrid dimensionality reduction technique	AC, DR FAR, precision, F-measure	Gained enhanced detection accuracy
[58]	Public data sets	IoT-based IDS	FPR and DR	Obtained satisfactory results
[59]	NSL-KDD	XGBoost–DNN model	Recall, accuracy, precision	Showed useful outcomes
[60]	UGR'16	ML-based NIDS	F1, precision, recall	Proved suitable for NID problems
[61]	NSL-KDD and UNSW-NB15	Hybrid IDS	DA, DR, FNR, FPR, FAR	0.01% FPR
[62]	N-BaIoT	CNN, RNN, and LSTM	F1	0.87 for 4 types of attack
[63]	PhishTank, OpenPhish, Curlie, N-BaIoT	Temporal LSTM	F1, accuracy	94.80% accuracy
[64]	N-BaIoT	ANN, J48, Naïve Bayes	Accuracy	99%
[65]	ISOT, ISCX	Artificial Immune System	Accuracy	0.99

Novel utilization of Genetic Algorithm (GA) alongside an immune algorithm was proposed in [47] to improve a computer's ability to detect intrusion. They carried out several simulations to verify the performance of their proposed method. The experiments conducted proved that Immune Genetic Algorithm (IGA) can enhance the system's capabilities to foretell the existence of an intrusion in the network.

The authors [48] used the M-AdaBoost-A algorithm for efficient intrusion detection in the network. They grouped several M-AdaBoost-A-based classifiers through several approaches such as PSO. Their proposed method came up with better performance than existing approaches among various classes in intrusion detection in the traditional enterprise and 802.11 wireless.

Another study [49] suggested a novel method for intrusion detection, namely SRDLM, based on deep learning and semantic re-encoding. Their proposed method enhances the classification abilities by utilizing deep learning while offering highly robust and accurate results. Their proposed method showed an accuracy of 99% when detecting the Web character injection network's attack. Injadat et al. [50] suggested multi-stage optimized Machine Learning-based NIDS that minimizes computations' complexity during intrusion detection. They compared correlation-based and information gain and studied their impact in terms of time and performance. Their proposed method was tested by using the UNSW-NB 2015 and CICIDS 2017 data sets.

A framework proposed in [51] used genetic algorithm (GA), firefly optimization (FFA), grey wolf optimizer (GWO), and particle swarm optimization (PSO). The framework was tested on UNSW-NB15 data set, J48 ML classifiers, and support vector machine (SVM) with promising results.

A study [52] proposed a novel framework based on DL and ML strategies to design NIDS. They studied the pros and cons of existing approaches to the suggested method. They aimed to facilitate researchers with advanced knowledge of AI-based NIDS and spotted possible obstacles for the proposed method.

Another study [53] proposed linear nearest neighbor lasso step (LNNLS-KH) to select features of intrusion detection. They implemented LNNLS-KH on renewed krill herd position to obtain the optimal global solution.

B et al. [54] applied wrapper and filter-based approaches using the firefly algorithm for feature selection. Classifiers of Bayesian Networks (BN) and C4.5 were applied to obtained features with a data set of KDD CUP 99. The preliminary outcomes proved that ten features are adequate for intrusion detection, providing enhanced accuracy.

A novel framework [40] named anomaly network-based IDS (A-NIDS) was proposed by utilizing AdaBoost and the artificial bee colony (ABC) algorithm to achieve low False Positive Rate (FPR) and high Detection Rate (DR). For the selection of features, they utilized ABC, and for classification and evaluation, they used the AdaBoost algorithm. The ISCX-IDS2012 and NSL-KDD data sets were used. Their framework showed promising results.

The authors of [55] proposed a method of intrusion detection for wireless networks using an improved convolutional neural network (ICNN). Firstly, they preprocessed and discriminated data and then model it using ICNN. The experimental results showed that their proposed method offers highly accurate results and valid favorable rates along with lower false-positive rates.

Another study [56] proposed an intrusion detection system based on anomaly to analyze and monitor network traffic flow towards a cloud system. For the classification of network traffic, they utilized an SVM classifier. To tune the SVM parameters, they used SPSO, and to select features of the network, they used binary-based Particle Swarm Optimization (BPSO). They used the NSL-KDD data set for the development and evaluation of their proposed method.

Salo et al. suggested a technique [57] of intrusion detection through principal component analysis (PCA) and information gain (IG). They utilized multi-layer perceptron (MLP), Instance-based learning algorithms (IBK), and support vector machine (SVM). They used

Kyoto 2006+, NSL-KDD, and ISCX 2012 to test their model. Their proposed method came up with highly accurate results.

A study [58] investigated network security issues, intrusion detection, fault tolerance, access control, authentication, key management, and crucial security technologies. It considered various intrusion detection approaches and their importance in IoT. They also made a comparison between various techniques of intrusion detection.

A method proposed in [59] used XGBoost to select features through a deep neural network (DNN) in the process of intrusion detection. The proposed model comprises classification, selection of features, and normalization, while for testing, the NSL-KDD data set was used.

Another study [60] analyzed and compared current machine learning-based NIDSs while using the UGR'16 data set to solve issues of intrusion detection. The guidelines provided will be helpful for researchers in analyzing NIDSs. Their proposed model still needs improvements and efforts to enhance understandability.

Krich et al. used stochastic optimization (SPSA) [66] to compute airborne weights that are low-side lobe beam-forming. Their proposed method only depends on digitizing the beam of radar's sum without demanding antenna calibration. The proposed approach comprises low-priced computations and can be scaled conveniently to radars as it contains a considerable quantity of elements of the antenna.

One study [61] suggested a novel method of hybrid classification based on Artificial Fish Swarm (AFS) and Artificial Bee Colony (ABC) algorithms. The authors utilized Correlation-based Feature Selection (CFS) and Fuzzy C-Means Clustering (FCM) to filter unnecessary features and to classify training data sets and evaluated their proposed model on the UNSW-NB15 and NSL-KDD data sets.

3. Material and Methods

3.1. Bat Algorithm

BA is a metaheuristic algorithm inspired by nature, proposed by Yang [67]. The microbats' property of echolocation is utilized in this algorithm used by microbats when searching for prey and handling hurdles faced in darkness. Microbats send quite loud pulses during searching prey, and these pulses start becoming quieter with an increase in distance.

Theories of the natural behavior of microbat are expressed as a method of optimization to propose BA. Every bat settles its velocity and location depending upon the mutual exclusivity of microbats close to prey to locate the accurate trajectory from their existing location, as illustrated in Figure 1.

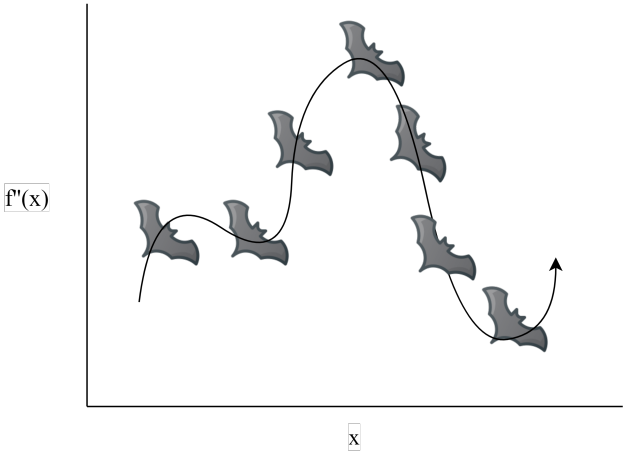


Figure 1. Trajectory of bats during the searching process.

To systematize the terms, Reference [67] proposed the following theories related to artificial bats.

1. Every micro bat estimates the distance within their surroundings and prey by utilizing its property of echolocation.
2. Frequency of fixed range is utilized to find a micro bat's velocity v'_i from location x'_i beside distinct loudness A'_o and distinct wavelength λ while searching for prey.
3. Emission pulse rate $r' \in [0, 1]$ can be utilized to adjust the frequency of its pulses while estimating distance among prey and microbat.
4. Loudness will be migrated from a considerable positive value A'_o to a smaller value A'_{min}.

Conventional BA comprises the following six steps:

Step 1: The bat parameters and the population are initialized. The global optimization function is expressed as follows:

$$min/\max\{f(x')|x' \in X\} \tag{1}$$

The objective function $f'(x)$ is utilized to estimate solution $x' = (x'_1, x'_2, ..., x'_d)$, the length demonstrated by d.

The following steps are followed for initialization of the BA parameters:

1. N denotes the number of locations of the artificial bat.
2. F'_{min} is used to denote minimum frequency, while F'_{max} is used to denote maximum frequency.
3. Every bat's velocity vector is represented by v'_j.
4. Rate of loudness is denoted by A'_j.
5. Pulse rate is denoted by r'_j.
6. Every bat's initial pulse rate is denoted by r'^0_j.
7. α and β are two constants.
8. £ is used to denote a range of bandwidth.

Step 2: Bat population memory is initialized. BM is used to stores vectors of bat location. The following steps are followed to generate those vectors randomly as expressed in Equation (2):

$$x_i^{'j} = LB_i + (UB_i - LB_i) \star (0, 1) \tag{2}$$

BM is then used to store those solutions while arranging values of objective function in ascending order.

$$BM = \begin{vmatrix} x'^1_1 & x'^1_2 & \dots & x'^1_d \\ x'^2_1 & x'^2_2 & \dots & x'^2_d \\ \vdots & \vdots & \dots & \vdots \\ x'^N_1 & x'^N_2 & \dots & x'^N_d \end{vmatrix} \tag{3}$$

Step 3: Regeneration of current population of bat.

Three operators are used to reconstruct every bat's location; these operators are selection, diversification, and intensification. While applying the intensification operator, a new location x'^j of the bat is created as follows:

$$F'_j = F'_{\min} + \left(F'_{\min} - F'_{\max}\right) * (0, 1) \tag{4}$$

$$v_i^{'j} = v_i^j + \left(x_i^{'j} - x_i^{'Gbest}\right) * F'_j \tag{5}$$

$$x_i^{''j} = x_i^{'j} + v_i^{'j} \tag{6}$$

While applying diversification operator, a local strategy of searching is used to create the new location of the bat. Consequently, the latest location x'^j is obtained as follows:

$$x_i^{''j} = x_i^{'best} + \in A'_j \tag{7}$$

$$x'^{j} \leftarrow \left\{ \begin{array}{cc} x'^{best} + \in A_j^{'} & U^{'}(0,1) > r_j' \\ x_j' + v_j' & \text{otherwise} \end{array} \right\} \tag{8}$$

While applying a selection operator, the current location of bat is replaced with a new location of bat and updates x'^{Gbest} in the case $f(x_i^{''j}) < f(x_i^{'Gbest})$.

$$r_j' = r_j'^{0}\left(1 - e'^{(-\gamma *)}\right) \tag{9}$$

$$A_j^{'} = \alpha A_j^{'} \tag{10}$$

$$A_j^{'itr} \to 0, r_j^{'itr} \to r_0^{'j}, whereitr \to \infty \tag{11}$$

Step 4: Stopping criteria.

The previous step is iterated until the criteria of termination are met. A pseudo code for standard BA is presented in Algorithm 1.

Algorithm 1 Pseudo code for standard BA.

```
Sensor with rich RSSI value
for j = 1 to N do
  for i = 1 to d do
    x_i^j = LB_i + (UB_i − LB_i) ⋆ (0,1)
  end for
end for
Calculate x'^Gbest, where Gbest ∈ (1,2,…..,N),
while itr<Total Iterations do
  for j = 1 to N do
    F'_j = F'_min + (F'_min − F'_max) ∗ (0,1)
    for i = 1 to d do
      v_i^j = v_i^j + (x_i^j − x_i^'Gbest) ∗ F'_j
      x_i^''j = x_i^'j + v_i^'j
    end for
    if U'(0,1) > r'_j then
      for i = 1 to d do
        x_i^''j = x_i^'best + ∈ A'_j
      end for
    end if
    if U'(0,1) > A'_j and f(x_i^''j) < f(x_i^'Gbest) then
      x'^j = x'^'j
        f'(x'^j) = f'(x'^'j)
      A'_j = α A'_j
      r'_j = r'_j^0 (1 − e'^(−γ*))
    end if
  end for
  Update x^'Gbest, Gbest ∈ (1,2,….., N)
end while
```

Deep Artificial Neural Networks (DNN)

An artificial neuron comprises interconnected processing units responsible for processing in parallel and assigns inputs to the required outputs. The output gained from the artificial neuron is illustrated as Equation (12). The DNN model consists of the following layers: output layer, hidden layers, and input layer.

$$y'_i = f'_i\left(\sum_{i=1}^{n} w'_{i.j} \cdot x'_i + \beta'_i\right) \tag{12}$$

Every neuron receives a signal from any environment. A weight $w'_{i.j}$ is linked with each input signal x'_i. The output signal y'_i is computed by the environment. In Equation (12), the node's output is denoted by y'_i, the node's ith input is denoted by x'_i, weight among input and node are denoted by w'_{ij}, the node's bias is denoted by β'_i, and the activation function of the node is expressed by f'_i. Generally, the node's activation function is a function of nonlinear nature, such as Gaussian function, sigmoid function, and Heaviside function. The output layer comprises a single neuron concerning each class.

To train neural networks by utilizing a meta-heuristic algorithm, three methods are available. In the first one, algorithms are utilized to gain a mixture of biases and weights that contribute the least MSE. Secondly, an appropriate structure for a given problem is found by utilizing algorithms. Thirdly, the gradient-based learning algorithm's parameters such as momentum and learning rate are adjusted by utilizing a meta-heuristic algorithm. Figure 2 demonstrates the DNN with one hidden layer.

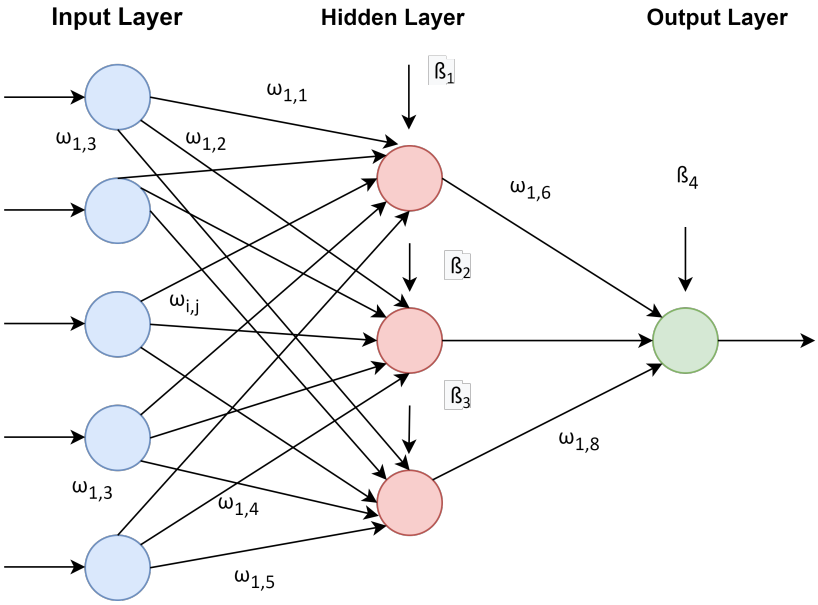


Figure 2. Sample neural network architecture with one hidden layer.

For swarm and evolutionary-based algorithms, the process of adaption is interpreted into an appropriate illustration of termination condition, DNN weights, and fitness function.

3.2. Local-Global Best Bat Algorithm (LGBA-NN)

Instant detection increases network security by speeding up alerts and by disconnecting compromised IoT devices from the network, preventing the botnet from spreading and preventing further attacks. In related work, several metaheuristics methods for detecting botnet attacks have been developed. The major problem in the metaheuristic algorithms used for botnet attack detection is premature convergence and low diversity.

A lack of diversity is usually the cause of premature convergence. The extent that changes, i.e., the variety of diverse solutions in the population and how specific they are, is measured by low diversity (distance between alternative solutions). Researchers have tried to propose new variants with the latest features in the last decade, but they all have the same advantages, such as achieving the global best optimum, and the same disadvantages, such as premature convergence and low diversity. To overcome this issue, we propose three modifications in the proposed LGBA-NN before use as an optimizer in the neural network.

The first modification includes robust population initialization of bats through Gaussian distribution. The proposed population initialization helps the bat's solution obtain enough divers for generating robust offsprings. Secondly, the proposed BA adopts the local-global best-based inertia weight to update the swarm's entire bat's velocity. Lastly, the local search mechanism is proposed and follows the Gaussian density function and local-global best function to achieve better exploration during each generation.

3.2.1. Gaussian Distribution

For a real-valued random variable, a Gaussian distribution is a form of the variable stochastic process. The probability density function's general form is as follows:

$$f(x) = \frac{1}{\rho\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(a-\sigma)^2}{\rho}} \tag{13}$$

In the above equation, the mean or expectation of the distribution (and its mode and median) is denoted by σ, while the standard deviation is shown as ρ. The distribution's variance is ρ^2. A normal deviate is a random variable that has a Gaussian distribution and is normally distributed.

A Gaussian function is as follows:

$$f(x) = x \times e^{\left(-\frac{(a-y)^2}{2z^2}\right)} \tag{14}$$

For nonzero z and arbitrary real constants x and y, a Gaussian's graph has a distinctive differential "bell curve" form. The intensity of the curve's edge is controlled by the parameter x, the direction of the peak's centre is controlled by the parameter y, and the width of the "bell" is controlled by the parameter z (the standard deviation, also known as the Gaussian Root Mean Square (RMS) width).

When combining an exponential function with a concave quadratic function, it forms a function:

$$f(x) = (\alpha + \beta y + \gamma) \tag{15}$$

where $\alpha = -0.5/c^2$, $\beta = b/c^2$ and $\gamma = \frac{0.5(Log(a-b^2)}{c^2})$

3.2.2. LGBA-NN Steps

The proposed architecture of LGBA-NN is presented in Figure 3.

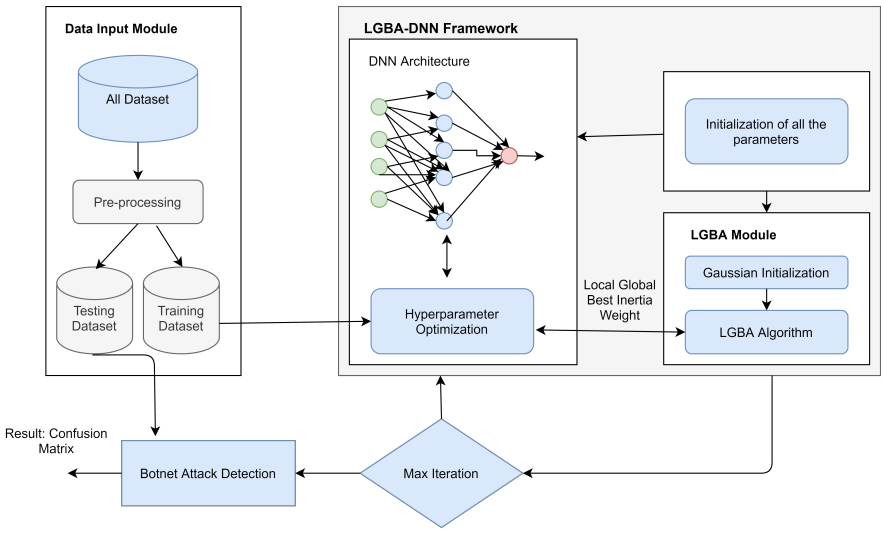


Figure 3. Proposed architecture of Local-Global best Bat Algorithm for Neural Network (LGBA-NN).

The following are the steps adopted in the proposed LGBA-NN.

1. Initialize the bat population using the Gaussian distribution over internal [0, 1]. The Gaussian density function ensures the diverse locations of each bat in the multidimensional search space. If any two corresponding vectors obtain the same initial solution, the Gaussian density function produces a diversity effect in the second generation. The following equation can be used to initialize the population using Gaussian distribution.

$$x_i^{'G} = LB_i + (UB_i - LB_i) * Gauss(0,1) \tag{16}$$

where $x_i^{'G}$ represents each individual gained initial location using Gaussian distribution over jth dimension. LB_i indicates the lower bound, which is set to 0, and UB_i shows the upper bound with 1 maximum value. $Gauss(0,1)$ represents the Gaussian density function used to generate random numbers following Gaussian distribution over the interval of [LB_i, UB_i]. Normal distribution starts with a random guess if the bats have any prior knowledge about the solution. However, Gaussian distribution disregards previous experience about the solution. This phenomenon tends to produce a rich diversity in the initial solution of the population.
2. The second modification includes the local-global best inertia weight to accelerate each bat's velocity during the exploration process. In standard BA, the velocity of each bat is updated without inertia weight acceleration. In general, significant inertia weight is recommended for the initial stages of the quest process to improve global exploration (searching for new areas). However, the inertia weight is decreased for local exploration in the later stages (fine-tuning the current search area).
The proposed local-global best inertia weight effect can be produced by taking a significant difference in local best solution of the current swarm with the global best solution of all multitude in the multidimensional search space. This phenomenon supports the convergence process to be robust enough in terms of exploration; if the current generation gained two same fitness values, the proposed inertia weight uses both solutions to decide the bat's next location towards the global optimum. Local-global best inertia weight can be defined using the following equation.

$$W^{LG} = \left(Gauss(0,1) * \frac{x_i^{'j}}{x_i^{'Gbest}} \right) \tag{17}$$

where $x_i^{'j}$ indicates the local best solution of current swarm and $x_i^{'Gbest}$ expresses the global best solution of all the swarms in the population.
LGBA-NN updates the bats velocity through following equation.

$$v_i^{'LG} = W^{LG} * v_i^j + \left(x_i^{'j} - x_i^{'Gbest} \right) * F_j^{'} \tag{18}$$

W^{LG} is local-global best inertia weight; v_i^j indicates the previous velocity of current individual, where $x_i^{'j}$ is current position of ith bat; and $x_i^{'Gbest}$ expresses the global best solution of all swarms in the population. F'_j shows frequency of the ith bat in the multi-dimensional search space.
3. In the third modification, we updated the local search mechanism, which uses the Gaussian density function and local-global best function to achieve better exploration during each generation. During the local searching process, each bat's current fitness was evaluated and compared with each bat's previous fitness; however, in standard BA, the local search mechanism missed the local and global best solution during the last generation. In the previous generation, if any bat has a global fitness greater than the global fitness of bats in the next generation, then the algorithm needs to retain the last global best fitness. We tackled this issue by introducing the Gaussian density

function and local-global best function to achieve better exploration during each generation. Local-global best function retained the difference between previous local and global best. LGBA will check the difference between local and global best fitness; check if it is minimal, smaller than the current fitness; and retains the best optimum solution. LGBA uses the following equation to update the local search manner.

$$x_i^{''LG} = Gauss(0,1) * v_i^{'LG} - x_i^{'best} \in 0.001 * A'_j \tag{19}$$

where $Gauss(0,1)$ represents the accelerated sequence generated through Gaussian distribution, $v_i^{'LG}$ indicates the local-global best function retaining the difference between the previous local and global best, where 0.001 is the scaling factor used to balance the exploitation during local search.

The pseudo code for proposed LGBA-NN is given in Algorithm 2.

4. In the last modification, LGBA is further employed in the neural network for hyperparameter tuning and weight optimization to classify ten different botnet attacks with an additional one benign target class. A weight $w'_{i.j}$ is linked with each input signal x'_i following the optimal solution x'^{LG}_i. The output signal y'_i is then fed to the classifier for the prediction. For weight optimization of LGBA-NN, we used the following equation.

$$w^{LG}{}_{i.j} = x_i^{''LG} + w'_{i.j} \tag{20}$$

Algorithm 2 Pseudo code for the proposed LGBA-NN.

Sensor with rich RSSI value
for $j = 1$ *to* N **do**
  for $i = 1$ *to* d **do**
    Use Equation (16) for population initialization
  end for
end for
Calculate x'^{Gbest}, *where* $Gbest \in (1, 2, \ldots..,N)$,
while itr<Total Iterations **do**
  for $j = 1$ *to* N **do**
    $F'_j = F'_{\min} + \left(F'_{\min} - F'_{\max}\right) * (0,1)$
    for $i = 1$ *to* d **do**
      Use Equation (18) to update velocity of each bat
      $x_i^{''j} = x_i^{'j} + v_i^{'j}$
    end for
    if $U'(0,1) > r'_j$ **then**
      for $i = 1$ *to* d **do**
        Use Equation (19) to obtain local best solution.
      end for
    end if
    if $U'(0,1) > A'_j$ and $f(x_i^{''j}) < f(x_i^{'Gbest})$ **then**
      $x'^j = x'^j$
      $f'(x'^j) = f'(x'^j)$
      $A'_j = \alpha A'_j$
      $r'_j = r'^0_j \left(1 - e'^{(-\gamma *)}\right)$
    end if
  end for
  Update x'^{Gbest}, $Gbest \in (1, 2, \ldots.., N)$
end while

${w^{LG}}_{i.j}$ shows the optimal weights obtained by LGBA and can further be used in the DNN as follows:

$$ {y'}_i = {f'}_i \left(\sum_{i=1}^{n} {w^{LG}}_{i.j} \, . \, {x'}_i + {\beta'}_i \right) \tag{21} $$

For hyperparameter optimization, LGBA-NN used x'^{LG}_i as an optimizer instead of other optimizers such as Adam and SGD.

3.3. Data Set Description

The proposed LGBA-NN was tested on an N-BaIoT data set with extensive real traffic data with benign and Malicious target classes. The N-BaIoT data set addresses the lack of publicly available botnet data sets, specifically for IoT technology. It shows accurate traffic data from 9 commercial IoT devices that have been infected with Gafgyt and Mirai. Gafgyt is considered one of the most well-known IoT botnets, and therefore, its script and activities have been replicated in other IoT malware. The botnet attacks IoT devices running on Linux by brute-forcing devices' default credentials while using open Telnet ports to launch an attack. N-BaIoT is a multivariate sequential data set with a total of 115 real value features. N-BaIoT is publically available at [68]. A data set description with a total number of target classes and the number of total instances in that class is given in Table 2.

Table 2. Data set description with the total number of target classes and the number of total instances.

Sr#	Target Class	Number of Instances
1	Mirai udpplain	15,304
2	Mirai udp	15,625
3	Mirai syn	16,436
4	Mirai scan	14,517
5	Mirai ack	15,138
6	Gafgyt udp	15,602
7	Gafgyt tcp	15,676
8	Gafgyt scan	14,648
9	Gafgyt junk	15,449
10	Gafgyt combo	15,345
11	Benign	15,538
12	**Total**	7,062,606

3.4. Evaluation Metrics

We used four performance measures to evaluate the performance of LGBA-NN. These evaluation metrics include precision, recall, f1-score, and support.

The recall is calculated by dividing the number of true positives by the number of false negatives plus true positives. True positives are independent variables that the LGBA-NN classifies as positive but are actually positive. While false negatives are independent variables that LGBA-NN classify as negative but are truly positive. Recall can be defined as follows:

$$ Recall = \frac{True\ Positives}{False\ Negatives + True\ Positives} \tag{22} $$

Precision is calculated by dividing the number of true positives by the number of false positives plus true positives. False positives are independent variables that the LGBA-NN classifies as positive but are actually negative. Precision can be defined as follows:

$$Precision = \frac{True\ Positives}{False\ Positives + True\ Positives} \tag{23}$$

F1 score is the harmonic mean of precision and recall:

$$F_1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{24}$$

4. Results and Discussion

Devices are usually equipped to classify based on expert labels in deep learning applications. LGBA-NN, on the other hand, has been trained to identify unusual behavior. As a result, LGBA-NN can detect previously unknown botnet behaviors, which is critical given the ever-evolving variants that render most detection methods obsolete. The IoT domain is too complicated compared to traditional computing environments. LGBA-NN, on the other hand, tackles the growing complexity of Smart nodes by evaluating each system against other botnet attack target groups. The traffic data of all linked hosts are supposed to be tracked in the enterprise scenario. Nonetheless, the volume of controlled traffic is too high to store and use for in-depth neural network training. To remove features, LGBA-NN employs systematic statistics. LGBA-NN training is available remotely. As a result, learning is useful and there is no need to be concerned about storage. Furthermore, since LGBA-NN is network-based, it consumes no computing memory from IoT devices typically constrained. As a result, LGBA-NN does not negatively impact its operation.

To the best of our knowledge, no one has previously applied optimized BA to an IoT network traffic for detecting ten botnet attacks. Therefore, we are still short on other variants to compare for a fair comparison of the state-of-the-art algorithms' current state. Furthermore, optimized BA has not been used as highly autonomous independent malware detectors in the broader domain of network traffic, instead of as intermediate devices with either object training or feature extraction or as semimanual anomaly indicators that rely heavily on human labeling for further examination by intelligence analysts.

We divided the experimental configuration into five phases. Firstly, the experiments were enriched with a neural network and no hyperparameter optimization; secondly, we added the Gaussian noise layer in the neural network to check the Gaussian effect on the classifier's performance. A performance evaluation of the neural network without hyperparameter optimization using ten attack types and one benign target class is given in Table 3. Similarly, a performance evaluation of the neural network without hyperparameter optimization and with Gaussian noise addition using ten attack types and one benign target class is presented in Table 4.

In the third phase, the Gaussian noise layer was extracted and the neural network dropout layer was added for further analysis of neural network behavior on the multi-class problem. Table 5, expressing the results, was obtained using the neural network without hyper-parameter optimization and with neural network layer dropout using ten attack types and one benign target class. After that, we merged both the Gaussian noise and neural network dropout layers without hyper-parameter optimization. Table 6 shows comparative results of the neural network without hyper-parameter optimization and with neural network layer dropout and Gaussian noise addition using ten attack types and one benign target class. Lastly, we evaluated LGBA-NN with hyper-parameter optimization. Since there are many other NN optimization methods proposed in the related work. Therefore, for a more thorough comparison, we included some other recent NN optimizers such as BA-NN and PSO-NN. The performance evaluations of BA-NN and PSO-NN using 10 attack types and 1 benign target class are presented in Tables 7 and 8. A

comparative evaluation of the proposed LGBA-NN using ten attack types and one benign target class is given in Table 9.

Table 3. Performance evaluation of the neural network without hyperparameter optimization using 10 attack types and 1 benign target class.

Attack Type	Precision	Recall	F1-Score	Support
Benign	0.9957	0.9724	0.9839	3882
Gafgyt_combo	0	0	0	3787
Gafgyt_junk	0.4977	0.9966	0.663898	3850
Gafgyt_scan	0.9975	0.9791	0.9882	3700
Gafgyt_tcp	0	0	0	3892
Gafgyt_udp	0.5089	0.9990	0.6743	4044
Mirai_ack	0.9974	0.3173	0.4814	3766
Mirai_scan	0.9806	0.9930	0.9868	3614
Mirai_syn	0.9885	0.9955	0.992068	4083
Mirai_udp	0.6545	0.9992	0.7909	3861
Mirai_udpplain	0.8714	0.9903	0.9271	3841
Average	0.6791	0.7510	0.6806	42,320

In terms of true positives, false negatives, true negatives, and false positives, for the most part, the LGBA-NN demonstrated dominance. Deep architectures' ability to learn variational structure representation and to estimate complex functions is likely to be the reason for this. The confusion matrix obtained through the neural network without hyperparameter optimization and with Gaussian noise addition using ten attack types and one benign target class is presented in Figure 4a.

Table 4. Performance evaluation of the neural network without hyperparameter optimization and with Gaussian noise addition using 10 attack types and 1 benign target class.

Attack Type	Precision	Recall	F1-Score	Support
Benign	0.9845	0.9976	0.9910	3828
Gafgyt_combo	0.5627	0.9513	0.7071	3846
Gafgyt_junk	0.8509	0.2729	0.4133	3913
Gafgyt_scan	0.9991	0.9848	0.9919	3636
Gafgyt_tcp	0.4997	1	0.666437	3867
Gafgyt_udp	0	0	0	3861
Mirai_ack	0.9946	0.9751	0.9847	3619
Mirai_scan	0.9913	0.9980	0.9947	3672
Mirai_syn	0.6814	0.9944	0.8087	4173
Mirai_udp	0.9783	0.9970	0.9875	4025
Mirai_udpplain	1	0.4961	0.663221	3880
Average	0.7733	0.7862	0.7432	42,320

LGBA-NN can minimize the loss of initiated attacks if the classification of attack-related abnormalities automatically and directly causes exclusion of the compromised IoT system from the network. Variation in loss during the training of neural networks without hyperparameter optimization is visualized in Figure 5a.

Furthermore, LGBA-NN cannot comprehend trivial identity mapping due to the restricted uncertainty imposed by the feature space in the hidden layers. As a result, LGBA-NN suits more common characteristics than unique ones. This is advantageous for IoT devices because their functionality usually is task-oriented, translating into a few standard traffic patterns. Variation in loss during the training of neural network without hyper-parameter optimization and with neural network layer dropout and LGBA-NN is visualized in Figure 5b,c. The confusion matrices obtained through a neural network without hyperparameter optimization and with neural network layer dropout, Gaussian noise addition, and LGBA-NN using ten attack types and one benign target class are illustrated in Table 9 and Figure 4c.

Table 5. Performance evaluation of the neural network without hyperparameter optimization and with neural network layer dropout using 10 attack types and 1 benign target class.

Attack Type	Precision	Recall	F1-Score	Support
Benign	0.9856	0.9977	0.9916	3919
Gafgyt_combo	0	0	0	3854
Gafgyt_junk	0.4959	0.9984	0.662748	3840
Gafgyt_scan	0.9991	0.9866	0.9928	3594
Gafgyt_tcp	0.4908	1	0.658513	3823
Gafgyt_udp	0	0	0	3954
Mirai_ack	0.9676	0.9674	0.9675	3805
Mirai_scan	0.9994	1	0.999724	3628
Mirai_syn	1	0.9990	0.999523	4192
Mirai_udp	0.9921	0.9687	0.9803	3907
Mirai_udpplain	0.9877	0.9994	0.9935	3804
Average	0.7175	0.8080	0.7472	42,320

Table 6. Performance evaluation of the neural network without hyperparameter optimization and with neural network layer dropout and Gaussian noise addition using 10 attack types and 1 benign target class.

Attack Type	Precision	Recall	F1-Score	Support
Benign	1	0.9770	0.988372	3828
Gafgyt_combo	0.7183	0.7771	0.7465	3846
Gafgyt_junk	0.7571	0.7009	0.7279	3913
Gafgyt_scan	0.9988	0.9898	0.994336	3636
Gafgyt_tcp	0.4994	0.9997	0.6660	3867
Gafgyt_udp	0	0	0	3861
Mirai_ack	0.8986	0.9361	0.917039	3619
Mirai_scan	0.9975	0.9989	0.9982	3672
Mirai_syn	0.9855	0.98130	0.9834	4173
Mirai_udp	0.9817	0.9085	0.9437	4025
Mirai_udpplain	0.9401	0.9994	0.9688	3880
Average	0.7973	0.8417	0.8114	42,320

Table 7. Performance evaluation of BA-NN using 10 attack types and 1 benign target class.

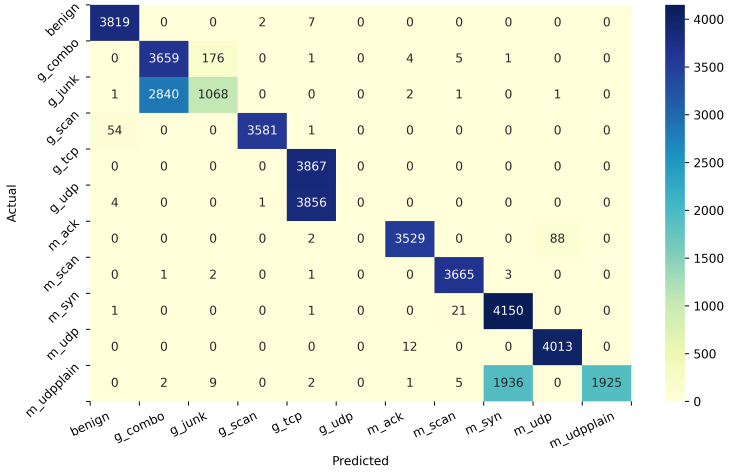
Attack Type	Precision	Recall	F1-Score	Support
Benign	0.987098	0.997699	0.99237	3911
Gafgyt_combo	0.989165	0.576276	0.72827	3802
Gafgyt_junk	0.705972	0.986229	0.822892	3776
Gafgyt_scan	0.942328	0.971896	0.956884	3665
Gafgyt_tcp	0	0	0	3903
Gafgyt_udp	0.498015	0.999486	0.664786	3891
Mirai_ack	0.967891	0.993147	0.980356	3794
Mirai_scan	0.974434	0.999727	0.986918	3660
Mirai_syn	0.979914	0.989377	0.984623	4142
Mirai_udp	0.999735	0.967444	0.983325	3901
Mirai_udpplain	0.993208	0.943484	0.967708	3875
Average	0.820524	0.855931	0.823409	42,320

Table 8. Performance evaluation of Particle Swarm Optimization (PSO)-NN using 10 attack types and 1 benign target class.

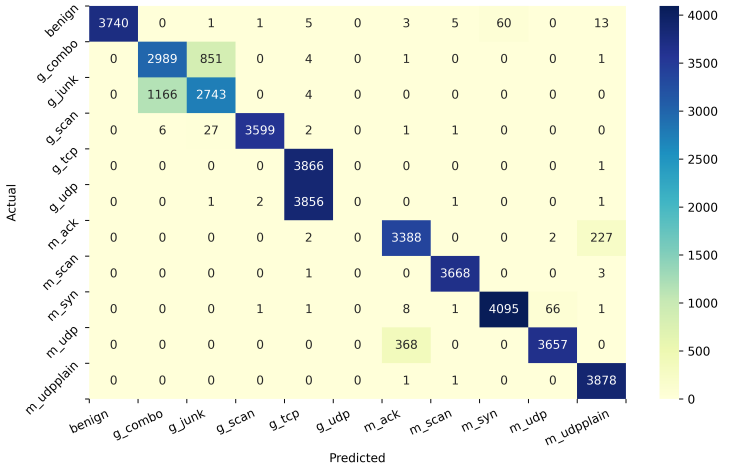
Attack Type	Precision	Recall	F1-Score	Support
Benign	0.999741	0.987983	0.993827	3911
Gafgyt_combo	0.989529	0.546817	0.704388	3802
Gafgyt_junk	0.692265	0.981197	0.811788	3776
Gafgyt_scan	0.998088	0.997271	0.99768	3665
Gafgyt_tcp	0.498722	0.999744	0.665473	3903
Gafgyt_udp	0	0	0	3891
Mirai_ack	0.925917	0.991566	0.957617	3794
Mirai_scan	0.955103	0.999727	0.976906	3660
Mirai_syn	0.970912	0.958957	0.964897	4142
Mirai_udp	0.997233	0.923866	0.959148	3901
Mirai_udpplain	0.989022	0.999742	0.994353	3875
Average	0.818607	0.852457	0.819568	42,320

Table 9. Performance evaluation of the proposed LGBA-NN using 10 attack types and 1 benign target class.

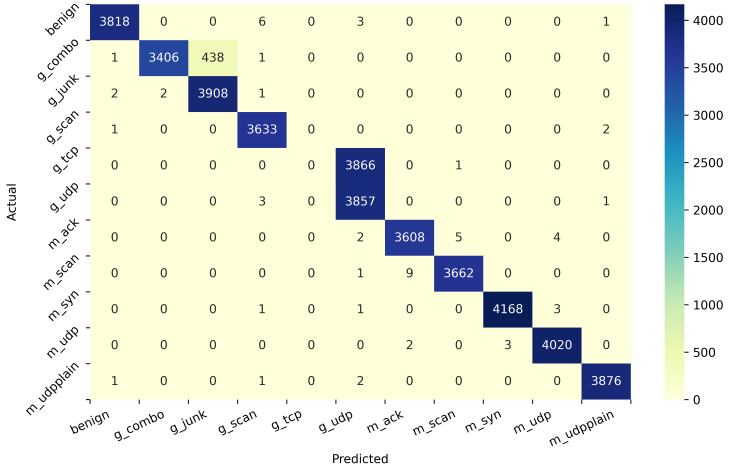
Attack Type	Precision	Recall	F1-Score	Support
Benign	0.9986	0.9973	0.9980	3828
Gafgyt_combo	0.9994	0.8855	0.9390	3846
Gafgyt_junk	0.8992	0.9987	0.9463	3913
Gafgyt_scan	0.9964	0.9991	0.9978	3636
Gafgyt_tcp	0	0	0	3867
Gafgyt_udp	0.4988	0.9989	0.6654	3861
Mirai_ack	0.9969	0.9969	0.99696	3619
Mirai_scan	0.9983	0.9972	0.9978	3672
Mirai_syn	0.9992	0.9988	0.9990	4173
Mirai_udp	0.9982	0.9987	0.9985	4025
Mirai_udpplain	0.9989	0.9989	0.9989	3880
Average	0.8523	0.9	0.8664	42,320



(**a**)

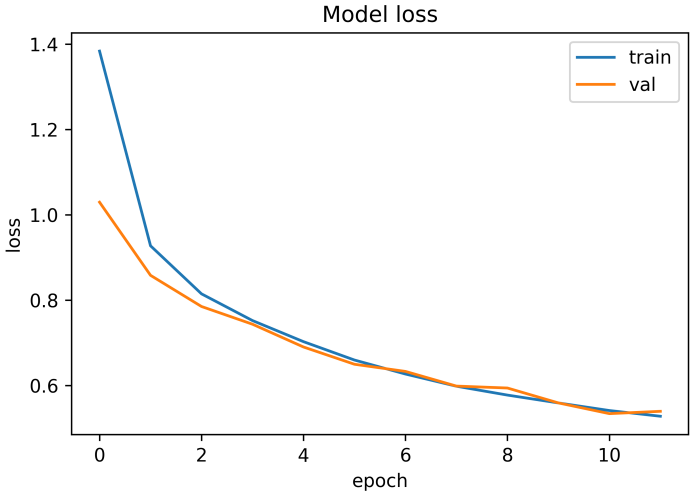


(**b**)

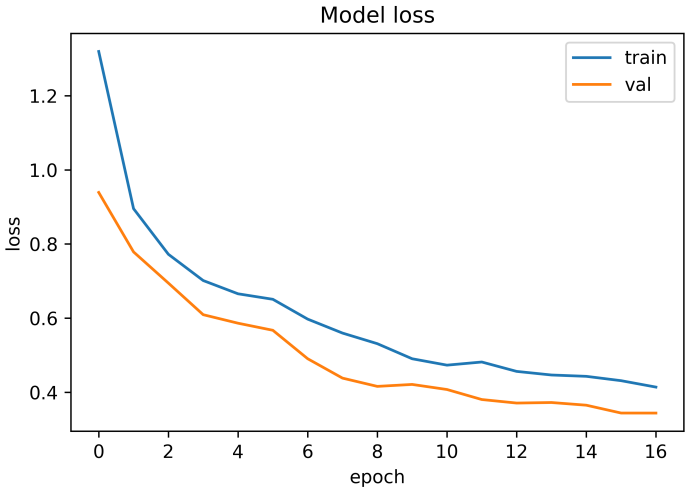


(**c**)

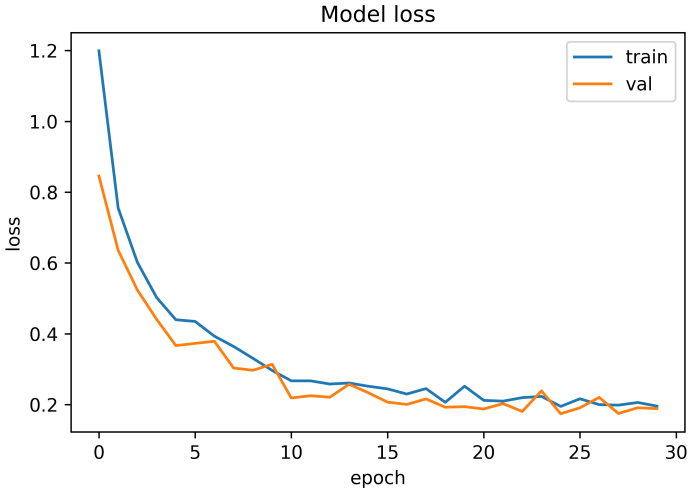
Figure 4. Confusion matrix of (**a**) the neural network without hyperparameter optimization and with Gaussian noise addition, (**b**) the neural network with hyperparameter optimization and with neural network layer dropout, and (**c**) the proposed LGBA-NN using 10 attack types and 1 benign target class.



(**a**)



(**b**)



(**c**)

Figure 5. Comparison of loss during training: (**a**) the neural network without hyperparameter optimization, (**b**) that without hyperparameter optimization and with neural network layer dropout, and (**c**) proposed LGBA-NN.

Analysis

Each type of botnet attack activity's uniformity can be applied appropriately to performance measures. An integrated platform with a high degree of traffic predictability can highlight any unusual behavior, increasing recall while decreasing precision. We extracted static and dynamic attributes from the training set for empirical validation, and we used NN to examine the impact of these attributes on the average recall and precision obtained by five NN configurations on the test set.

From Table 3, it can be observed that the neural network shows a higher precision rate of 0.9957 and a higher recall rate of 0.9724 for the benign class. However, regarding botnet attacks, identification neural network fails to deliver a low false-positive rate. Standard CNN without hyperparameter optimization obtained a low precision rate, which means the classifier identified both types of botnet attacks with ten target classes as benign instances at a higher rate. Similarly, a higher mean recall rate recorded for neural networks without hyperparameter optimization shows the ability of the neural network to classify benign samples as malicious botnet attacks, hence decreasing the false-negative rate.

Adding Gaussian noise layer to the neural network without hyperparameter optimization using LGBA slightly improved the average precision rate (see Table 4). However, the average recall rate's net impact compared to the neural network without the Gaussian noise layer is the same. This variant obtained a maximum precision rate of 0.9974 and a minimum recall rate of 0.3173 for the "Mirai ack" target class. As opposed to this, a neural network with a Gaussian noise layer failed to gain a low false-positive rate for the "Gafgyt udp" target class with a minimum precision rate of 0.5089 and a maximum recall rate of 0.9990.

Similarly, removing the Gaussian noise layer and adding a neural network dropout layer to the neural network without hyperparameter optimization using LGBA insignificantly increased the average precision rate (referred to Table 5). However, the average recall rate's net impact correlated to the neural network with the Gaussian noise layer is identical. This modification achieved a maximum precision rate of 1.00 and a maximum recall rate of 0.9990 for the "Mirai syn" target classes. This indicates that the false-positive and false-negative rates are approximately 0 for that particular target class. As opposed to this, adding a neural network dropout layer failed to gain a low false-positive rate for the "Gafgyt udp" and " Gafgyt combo" target classes with a minimum precision rate of 0.00 and minimum recall rate of 0.00.

From Table 6, it can be observed that a neural network without hyperparameter optimization and with neural network layer dropout and Gaussian noise addition bestows a more maximum precision rate of 0.9994 and a higher recall rate of 1.00 for the "Mirai scan" class. Nevertheless, concerning "Gafgyt udp" and "Gafgyt combo," the variant fails to deliver a low false-positive rate.

From Table 7, it can be perceived that the BA-NN confers a higher average recall rate of 0.855931 and a lower precision rate of 0.820524. However, BA-NN failed to obtain a low false-positive rate for the "Gafgyt tcp" target class with a minimum precision rate of 0.00 and a minimum recall rate of 0.00. Referring to Table 8, PSO-NN obtained a high precision rate for all target classes except "Gafgyt tcp" and "Gafgyt udp" with 0.498 and 0.00 respectively. However, both variants BA-NN and PSO-NN achieved higher accuracy compared to the non-optimziation version of the experimental results.

The proposed LGBA-NN (refer to Table 9) managed to overcome the effects of each botnet attack detection except the "Gafgyt tcp" botnet attack class. We can observe that LGBA-NN obtained a low false-negative rate as it misclassified only 10% of its negative instances. However, it still had a 15% false-positive rate, indicating the complexity of multi-class dimensionality. The proposed LGBA-NN received a maximum precision rate of 0.998969 and a maximum recall rate of 0.9989 for the "Mirai udpplain" target classes. Compared to all neural networks, LGBA-NN shows a maximum accuracy of 90%, with the lowest misclassification rate for all target classes. The loss curves and confusion matrix presented in Figures 4 and 5c also confirm the superiority of the proposed LGBA-NN over

other variants of the neural network, as LGBA-NN (90% accuracy) outperformed BA-NN (85.5% accuracy) and PSO-NN (85.2% accuracy).

5. Conclusions

To reduce the risk associated with IoT devices, it is essential to identify DDoS attacks in advance. Early DDoS attack identification improves network security by speeding up the process of disconnecting compromised IoT devices from the network, preventing the botnet from spreading and preventing additional attacks. In this research, LGBA-NN was proposed to accumulate both feature subsets and hyperparameters for efficient botnet detection based on data from 9 commercial IoT devices that were authentically infected by two botnets: Gafgyt and Mirai. The proposed BA uses local-global best-based inertia weight to update the swarm's entire bat's velocity. To tackle with swarm diversity of BA, we proposed Gaussian distribution used in the population initialization. Furthermore, the local search mechanism was enhanced by the Gaussian density function and local-global best function to achieve better exploration during each generation. The proposed LGBA-NN was put to the test on the N-BaIoT data set, which includes a large amount of real-time traffic data for both benign and malicious target groups. We evaluated the proposed LGBA-NN by comparing the performance with several configurations of a non-optimized version of neural networks and some recent variants of optimized neural networks such as PSO-NN and BA-NN. The experimental results proved that LGBA-NN is superior over other recent algorithms.

In future work, we intend to extend the optimization of neural networks using the bat algorithm to other evolutionary models such as differential evolution algorithm, genetic algorithm, and particle swarm optimization.

Author Contributions: Conceptualization, A.A. and W.A.; methodology, A.A., W.A., and H.A.; software, H.T.R.; validation, H.T.R. and R.D.; writing—original draft preparation, A.A., W.A., and H.A.; writing—review and editing, H.T.R. and R.D.; visualization, H.T.R.; supervision, R.D.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Taif University Researchers Supporting Project under grant number TURSP- 2020/231, Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data set used for this study is publicly available at [68].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vasilomanolakis, E.; Karuppayah, S.; Mühlhäuser, M.; Fischer, M. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Comput. Surv.* **2015**, *47*, 1–33. [CrossRef]
2. Hussain, B.; Du, Q.; Sun, B.; Han, Z. Deep Learning-Based DDoS-Attack Detection for Cyber-Physical System over 5G Network. *IEEE Trans. Ind. Inform.* **2021**, *17*, 860–870. [CrossRef]
3. de Assis, M.V.O.; Carvalho, L.F.; Rodrigues, J.J.P.C.; Lloret, J.; Proença, M.L., Jr. Near real-time security system applied to SDN environments in IoT networks using convolutional neural network. *Comput. Electr. Eng.* **2020**, *86*, 106738. [CrossRef]
4. Manimurugan, S.; Al-Mutairi, S.; Aborokbah, M.M.; Chilamkurti, N.; Ganesan, S.; Patan, R. Effective attack detection in internet of medical things smart environment using a deep belief neural network. *IEEE Access* **2020**, *8*, 77396–77404. [CrossRef]
5. Rehman Javed, A.; Jalil, Z.; Atif Moqurrab, S.; Abbas, S.; Liu, X. Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles. *Trans. Emerg. Telecommun. Technol.* **2020**. [CrossRef]
6. Lee, S.; Abdullah, A.; Jhanjhi, N.; Kok, S. Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning. *PeerJ Comput. Sci.* **2021**, *7*, 1–23. [CrossRef]
7. Perez, M.G.; Celdran, A.H.; Ippoliti, F.; Giardina, P.G.; Bernini, G.; Alaez, R.M.; Chirivella-Perez, E.; Clemente, F.J.G.; Perez, G.M.; Kraja, E.; et al. Dynamic Reconfiguration in 5G Mobile Networks to Proactively Detect and Mitigate Botnets. *IEEE Internet Comput.* **2017**, *21*, 28–36. [CrossRef]
8. Wei, W.; Woźniak, M.; Damaševičius, R.; Fan, X.; Li, Y. Algorithm Research of Known-plaintext Attack on Double Random Phase Mask Based on WSNs. *J. Internet Technol.* **2019**, *20*, 39–48.

9. Yong, B.; Wei, W.; Li, K.; Shen, J.; Zhou, Q.; Wozniak, M.; Połap, D.; Damaševičius, R. Ensemble machine learning approaches for webshell detection in Internet of things environments. *Trans. Emerg. Telecommun. Technol.* **2020**. [CrossRef]
10. Chung, Y.Y.; Wahid, N. A hybrid network intrusion detection system using simplified swarm optimization (SSO). *Appl. Soft Comput.* **2012**, *12*, 3014–3022. [CrossRef]
11. Ganapathy, S.; Kulothungan, K.; Muthurajkumar, S.; Vijayalakshmi, M.; Yogesh, P.; Kannan, A. Intelligent feature selection and classification techniques for intrusion detection in networks: A survey. *EURASIP J. Wirel. Commun. Netw.* **2013**, *2013*, 1–16. [CrossRef]
12. Aburomman, A.A.; Reaz, M.B.I. Review of IDS development methods in machine learning. *Int. J. Electr. Comput. Eng. (IJECE)* **2016**, *6*, 2432–2436. [CrossRef]
13. Bijalwan, A. Botnet Forensic Analysis Using Machine Learning. *Secur. Commun. Netw.* **2020**, *2020*, 9302318. [CrossRef]
14. Alothman, Z.; Alkasassbeh, M.; Al-Haj Baddar, S. An efficient approach to detect IoT botnet attacks using machine learning. *J. High Speed Netw.* **2020**, *26*, 241–254. [CrossRef]
15. Damasevicius, R.; Venckauskas, A.; Grigaliunas, S.; Toldinas, J.; Morkevicius, N.; Aleliunas, T.; Smuikys, P. Litnet-2020: An annotated real-world network flow dataset for network intrusion detection. *Electronics* **2020**, *9*, 800. [CrossRef]
16. Mahmood, T.; Afzal, U. Security analytics: Big data analytics for cybersecurity: A review of trends, techniques and tools. In Proceedings of the 2013 2nd National Conference on Information Assurance (NCIA), Rawalpindi, Pakistan, 11–12 December 2013; pp. 129–134.
17. Cozzi, E.; Vervier, P.A.; Dell'Amico, M.; Shen, Y.; Bilge, L.; Balzarotti, D. The Tangled Genealogy of IoT Malware. In Proceedings of the Annual Computer Security Applications Conference, Austin, TX, USA, 7–11 December 2020. [CrossRef]
18. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
19. Hoque, N.; Bhattacharyya, D.K.; Kalita, J.K. Botnet in DDoS Attacks: Trends and Challenges. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2242–2270. [CrossRef]
20. McDermott, C.D.; Majdani, F.; Petrovski, A.V. Botnet Detection in the Internet of Things using Deep Learning Approaches. In Proceedings of the International Joint Conference on Neural Networks, Rio de Janeiro, Brazil, 8–13 July 2018; Volume 2018.
21. Koroniotis, N.; Moustafa, N.; Sitnikova, E. Forensics and Deep Learning Mechanisms for Botnets in Internet of Things: A Survey of Challenges and Solutions. *IEEE Access* **2019**, *7*, 61764–61785. [CrossRef]
22. Yerima, S.Y.; Alzaylaee, M.K.; Shajan, A.; Vinod, P. Deep learning techniques for android botnet detection. *Electronics* **2021**, *10*, 519. [CrossRef]
23. Marir, N.; Wang, H.; Feng, G.; Li, B.; Jia, M. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access* **2018**, *6*, 59657–59671. [CrossRef]
24. Azeez, N.A.; Ayemobola, T.J.; Misra, S.; Maskeliūnas, R.; Damaševičius, R. Network intrusion detection with a hashing based apriori algorithm using Hadoop MapReduce. *Computers* **2019**, *8*, 86. [CrossRef]
25. Tuan, T.A.; Long, H.V.; Son, L.H.; Kumar, R.; Priyadarshini, I.; Son, N.T.K. Performance evaluation of Botnet DDoS attack detection using machine learning. *Evol. Intell.* **2020**, *13*, 283–294. [CrossRef]
26. Kebande, V.R.; Venter, H.S. A cognitive approach for botnet detection using Artificial Immune System in the cloud. In Proceedings of the 2014 Third International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), Beirut, Lebanon, 29 April–1 May 2014; pp. 52–57.
27. Da, K. A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
28. Zeiler, M.D. Adadelta: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
29. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
30. Rauf, H.T.; Malik, S.; Shoaib, U.; Irfan, M.N.; Lali, M.I. Adaptive inertia weight Bat algorithm with Sugeno-Function fuzzy search. *Appl. Soft Comput.* **2020**, *90*, 106159. [CrossRef]
31. Ullah, I.; Mahmoud, Q.H. A two-level flow-based anomalous activity detection system for IoT networks. *Electronics* **2020**, *9*, 530. [CrossRef]
32. Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; pp. 581–585.
33. Folorunso, O.; Ayo, F.E.; Babalola, Y. Ca-NIDS: A network intrusion detection system using combinatorial algorithm approach. *J. Inf. Priv. Secur.* **2016**, *12*, 181–196. [CrossRef]
34. Deng, L. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Trans. Signal Inf. Process.* **2014**, *3*, e2. [CrossRef]
35. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A survey of deep learning methods for cyber security. *Information* **2019**, *10*, 122. [CrossRef]
36. Yilmaz, S.; Sen, S. Early Detection of Botnet Activities Using Grammatical Evolution. In *Applications of Evolutionary Computation*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 395–404. [CrossRef]
37. Yu, Y.; Long, J.; Liu, F.; Cai, Z. Machine learning combining with visualization for intrusion detection: A survey. In Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence, Sant Julià de Lòria, Andorra, 19–21 September 2016; pp. 239–249.

38. Ahmed, A.A.; Jabbar, W.A.; Sadiq, A.S.; Patel, H. Deep learning-based classification model for botnet attack detection. *J. Ambient. Intell. Humaniz. Comput.* **2020**. [CrossRef]
39. Alauthman, M.; Aslam, N.; Al-kasassbeh, M.; Khan, S.; Al-Qerem, A.; Raymond Choo, K. An efficient reinforcement learning-based Botnet detection approach. *J. Netw. Comput. Appl.* **2020**, *150*, 102479. [CrossRef]
40. Mazini, M.; Shirazi, B.; Mahdavi, I. Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *J. King Saud Univ. Comput. Inf. Sci.* **2019**, *31*, 541–553. [CrossRef]
41. Asadi, M.; Jabraeil Jamali, M.A.; Parsa, S.; Majidnezhad, V. Detecting botnet by using particle swarm optimization algorithm based on voting system. *Future Gener. Comput. Syst.* **2020**, *107*, 95–111. [CrossRef]
42. 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. Humaniz. Comput.* **2020**, *11*, 2809–2825. [CrossRef]
43. Lin, K.C.; Chen, S.Y.; Hung, J.C. Botnet Detection Using Support Vector Machines with Artificial Fish Swarm Algorithm. *J. Appl. Math.* **2014**, *2014*, 1–9. [CrossRef]
44. Rajagopal, S.; Kundapur, P.P.; Hareesha, K.S. A stacking ensemble for network intrusion detection using heterogeneous datasets. *Secur. Commun. Netw.* **2020**, *2020*, 4586875. [CrossRef]
45. Elmasry, W.; Akbulut, A.; Zaim, A.H. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Comput. Netw.* **2020**, *168*, 107042. [CrossRef]
46. Dwivedi, S.; Vardhan, M.; Tripathi, S. Defense against distributed DoS attack detection by using intelligent evolutionary algorithm. *Int. J. Comput. Appl.* **2020**, 1–11. [CrossRef]
47. Suhaimi, H.; Suliman, S.I.; Musirin, I.; Harun, A.; Mohamad, R.; Kassim, M.; Shahbudin, S. Network intrusion detection system using immune-genetic algorithm (IGA). *Indones. J. Electr. Eng. Comput. Sci.* **2020**, *17*, 1059–1065. [CrossRef]
48. Zhou, Y.; Mazzuchi, T.A.; Sarkani, S. M-AdaBoost—A based ensemble system for network intrusion detection. *Expert Syst. Appl.* **2020**, *162*, 113864. [CrossRef]
49. Wu, Z.; Wang, J.; Hu, L.; Zhang, Z.; Wu, H. A network intrusion detection method based on semantic re-encoding and deep learning. *J. Netw. Comput. Appl.* **2020**, *164*, 102688. [CrossRef]
50. Injadat, M.; Moubayed, A.; Nassif, A.B.; Shami, A. Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection. *IEEE Trans. Netw. Serv. Manag.* **2020**. [CrossRef]
51. Almomani, O. A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms. *Symmetry* **2020**, *12*, 1046. [CrossRef]
52. Ahmad, Z.; Khan, A.S.; Shiang, C.W.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2020**, *32*. [CrossRef]
53. Li, X.; Yi, P.; Wei, W.; Jiang, Y.; Tian, L. LNNLS-KH: A Feature Selection Method for Network Intrusion Detection. *Secur. Commun. Netw.* **2021**, *2021*, 8830431 . [CrossRef]
54. Selvakumar, B.; Muneeswaran, K. Firefly algorithm based feature selection for network intrusion detection. *Comput. Secur.* **2019**, *81*, 148–155. [CrossRef]
55. Dong, Q.L.; He, S.N. Self-adaptive projection algorithms for solving the split equality problems. *Fixed Point Theory* **2017**, *18*, 191–202. [CrossRef]
56. Sakr, M.M.; Tawfeeq, M.A.; El-Sisi, A.B. Network Intrusion Detection System based PSO-SVM for Cloud Computing. *Int. J. Comput. Netw. Inf. Secur.* **2019**, *11*, 22–29. [CrossRef]
57. Salo, F.; Nassif, A.B.; Essex, A. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Comput. Netw.* **2019**, *148*, 164–175. [CrossRef]
58. Deng, L.; Li, D.; Yao, X.; Wang, H. RETRACTED ARTICLE: Mobile network intrusion detection for IoT system based on transfer learning algorithm. *Clust. Comput.* **2018**, *22*, 9889–9904. [CrossRef]
59. Devan, P.; Khare, N. An efficient XGBoost–DNN-based classification model for network intrusion detection system. *Neural Comput. Appl.* **2020**, *32*, 12499–12514. [CrossRef]
60. Magán-Carrión, R.; Urda, D.; Díaz-Cano, I.; Dorronsoro, B. Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches. *Appl. Sci.* **2020**, *10*, 1775. [CrossRef]
61. Hajisalem, V.; Babaie, S. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Comput. Netw.* **2018**, *136*, 37–50. [CrossRef]
62. Kim, J.; Shim, M.; Hong, S.; Shin, Y.; Choi, E. Intelligent detection of iot botnets using machine learning and deep learning. *Appl. Sci.* **2020**, *10*, 7009. [CrossRef]
63. De La Torre Parra, G.; Rad, P.; Choo, K.R.; Beebe, N. Detecting Internet of Things attacks using distributed deep learning. *J. Netw. Comput. Appl.* **2020**, *163*, 102662. [CrossRef]
64. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Machine learning-based IoT-botnet attack detection with sequential architecture. *Sensors* **2020**, *20*, 4372. [CrossRef]
65. Hosseini, S.; Nezhad, A.E.; Seilani, H. Botnet detection using negative selection algorithm, convolution neural network and classification methods. *Evol. Syst.* **2021**. [CrossRef]
66. Krich, S.I.; Weiner, I. Low-Sidelobe Antenna Beamforming Via Stochastic Optimization. *IEEE Trans. Antennas Propag.* **2014**, *62*, 6482–6486. [CrossRef]

67. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. In Proceedings of the Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Granada, Spain, 12–14 May 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74. [CrossRef]
68. 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]