Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 2039217, 16 pages
https://doi.org/10.1155/2023/2039217

Research Article

Optimization Enabled Deep Learning-Based DDoS Attack Detection in Cloud Computing

S. Balasubramaniam,[1] **C. Vijesh Joe**,[2] **T. A. Sivakumar**,[3] **A. Prasanth**,[4] **K. Satheesh Kumar**,[1] **V. Kavitha**,[5] **and Rajesh Kumar Dhanaraj**[6]

[1]*Department of Futures Studies, University of Kerala, Thiruvananthapuram, Kerala, India*
[2]*School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India*
[3]*Faculty of Engineering and Technology, Villa College, Male', Maldives*
[4]*Department of ECE, Sri Venkateswara College of Engineering, Sriperumbudur, Tamilnadu, India*
[5]*Department of Computer Science and Engineering, University College of Engineering, Kanchipuram, Tamil Nadu, India*
[6]*Department of Computer Science and Engineering, Galgotias University, Greater Noida, Utter Pradesh, India*

Correspondence should be addressed to S. Balasubramaniam; baluttn@gmail.com and T. A. Sivakumar; sivakumar.thankaraj@villacollege.edu.mv

Received 8 November 2022; Revised 1 February 2023; Accepted 4 February 2023; Published 20 February 2023

Academic Editor: Lianyong Qi

Cloud computing is a vast revolution in information technology (IT) that inhibits scalable and virtualized sources to end users with low infrastructure cost and maintenance. They also have much flexibility and these resources are supervised by various management organizations and provided over the Internet by known standards, formats, and networking protocols. Legacy protocols and underlying technologies consist of vulnerabilities and bugs which open doors for intrusion by network attackers. Attacks as distributed denial of service (DDoS) are one of most frequent attacks, which impose heavy damage and affect performance of the cloud. In this research work, DDoS attack detection is easily identified in an optimized way through a novel algorithm, namely, the proposed gradient hybrid leader optimization (GHLBO) algorithm. This optimized algorithm is responsible to train a deep stacked autoencoder (DSA) that detects the attack in an efficient manner. Here, fusion of features is carried out by deep maxout network (DMN) with an overlap coefficient, and augmentation of data is carried out by the oversampling process. Furthermore, the proposed GHLBO is generated by integrating the gradient descent and hybrid leader-based optimization (HLBO) algorithm. Also, this proposed method is assessed by various performance metrics, such as the true positive rate (TPR), true negative rate (TNR), and testing accuracy with values attained as 0.909, 0.909, and 0.917, accordingly.

1. Introduction

Cloud computing is an Internet-enabled platform for delivering computing facilities, including networking, servers, and databases to users or employers in organizations at huge scale, and helps companies with cost reduction for a particular organization [1]. Nowadays, cloud computing is growing as the standard platform for distributing large data pool that provides various user-friendly features. Most services related to cloud computing are of pay on demand type in which each and every user is allocated by discrete pool of devices used for data mining. Services of cloud computing are classified as IaaS, SaaS, and PaaS [2]. Cloud computing helps organizations or users to reduce the cost of infrastructure by supplementing various online resources that are in the form of services. In cloud computing, organizations or users pay only for service time based on duration in accordance to the pay-as-you-use policy. This service availability is very important and beneficial to users or organizations; else they have to tolerate big financial issues with or without reputation loss [3]. Default keys are used by cloud devices that have no roles on security on

acoustics which make them susceptible for negotiation. Cloud system contamination is frequently ignored by the user, and without proper awareness of owners in service, hundreds to thousands of devices are theoretically mistreated by the attacker for large-scale attack [4]. Technology advancement also has serious issues in the cloud and one of these threats is DoS. DoS creates unavailability of network services; however, this unavailability of services is the result of various other reasons, such as faults in software or cloud component [5].

DDoS attack is a common category of cyber-attack, which creates unauthorized and disturbed services to network users [6] that is utilized by attackers to avoid authentic users from retrieving services [1, 7]. Attackers use these DDoS attacks not to be available for authentic users [8]. Here, attackers put heavy load on network services provided by target server on public. Network known botnet of numerous hosts in Internet is used for distributing traffic to victim or user. Amplification and reflection techniques lead this DDoS attack to a much destructive state [3]. These attacks are carried out by compromising and exploiting hundreds to thousands of hosts, termed zombies, which execute attack against the machine of target. They disturb regular and normal traffic on a network via sudden exponential upsurge in traffic and lastly prevent regular traffic from attaining its terminus. DDoS is considered as a type of malicious attack on cloud servers that creates many severe problems [9]. These attacks generate large network traffic containing packets sent on the network, making regular users in trouble who want to obtain services that not respond to their requirements [6, 7]. Packets are categorized as normal or malicious based on DDoS defense methods, and these methods fall under two major types, (1) the signature-based method and (2) the anomaly-based method. Signature-based methods use many attack signatures situated in the knowledge database to detect attacks and effectively find known attacks. In the meantime, anomaly-based techniques analyse regular normal traffic behavioural patterns in a particular period for detecting deviation in the steady action and analyse the zero-day attack [2].

The DL system is very efficient in discriminating traffic of DDoS from benign traffic by extracting representations of traffic of the high-level feature from traffic of the low level [10]. Efficient disposition of technologies under security, including access control, cloud encryption, malware identification, and secure uploading is achieved by DL and computers [4]. It is suited for modelling a nonlinear complex relationship by learning various stages of representation that correspond to multiple stages of abstraction. DNN has a cascade of multiple layers of processing units, which is nonlinear for transformation and extracting features, that is, a promising technique for identifying attacks in social network [11]. Detecting a cyber-attack shares the feature that is common with the recognition of image, which harnessed new features of DL. Small changes in the pixel tend to identify image changes where attack is detected in the same way as more than 99 percentage of novel attacks are minor mutants of previous attacks. This reinforces efficiency of DL for detecting minor changes in patterns of attacks [12]. Unsupervised SA in DL learns representations from an unbalanced dataset that uses DT as the binary classifier for detecting attacks from newly merged representations [13]. DL is applied to cyber security because of the capability of self-learning and analyzing. Web attack detection within URLs from attackers and normal users by DL is a challenging task, and major problems include the following: (i) an effective way for transforming every kind of URL into representations is very important in view of multiple ways as various attacks hide in respective URLs, (ii) various attacks show various signatures in URLs, and thus selecting a feature is not much easy, and (iii) most DL applications in cyber security have one model to do detection, and it is difficult to update the system [14].

This work is concentrated in detecting DDoS attacks in cloud computing using the DL method, trained by an optimized algorithm. Here, the proposed optimization algorithm is named as GHLBO, which is generated by incorporating gradient descent with HLBO algorithm. Different stages involved for model detection are feature fusion, data augmentation, and finally attack detection. Here, process of fusion of features is carried out by DMN using an overlap coefficient, which is then followed by using data augmentation carried out by oversampling. Next to augmentation of data, the DDoS attack is detected by DSA that is trained by proposed GHLBO.

The main contributions of this article are as follows:

(i) developed GHLBO algorithm enabled DSA: estimating a DDoS attack is carried out using designed GHLBO, created by the collaboration of the gradient descent and HLBO algorithm. This GHLBO trains DSA for estimating or detecting a DDoS attack in cloud computing.

The remaining parts of this article include the following: Section 2 represents the literature review of attack detection and Section 3 represents the elaborate particulars of the proposed GHLBO-based DSA. Section 5 represents discussions with results of the developed model, and this article is concluded in Section 6.

2. Motivation

DDoS attack detection is much needed for helping the legitimate users to carefully access to network services. Multiple techniques are available for this detection; but those techniques are hard to trace back to attacker and not effective to mitigate these attacks. To overcome these problems, there is a need to adopt a best detection method. Hence, this proposed GHLBO-enabled DSA represents an optimal way for DDoS detection. This section also enhances literature reviews regarding current detection techniques along with uses, drawbacks, and challenges.

2.1. Literature Assessment. Assessment of reviews from literature of various researchers regarding DDoS detection in cloud computing is given as follows: Velliangiri et al. [2] proposed TEHO-enabled DBN, which was used to identify

attacks at earlier stages itself. But this method followed more iterations, TEHO-DBN for updating weights of input and hidden units of the MLP layer that tend to have more computational time. This drawback was overcome by Arul and Punidha [4], where SD-LVQ was developed; here, the cloud-mounted computer function was evaluated to reduce detection strategies of the DDoS-encrypted cross-site attack. However, the challenge by deep-supervised methods over the hybrid cloud data centre remained. Challenge in [4] was eradicated in [10]. Doriguzzi-Corin et al. [10] designed LUCID model architecture, which followed the lightweight application with less overhead processing and minimal time of detection. But time of convergence and accuracy was low in this method. This low convergence was removed in [15]. Agarwal et al. [15] developed FS-WOA, in which DDoS attack entry in the big-scale industry was avoided. However, this method lacks in generating individual instantiations to detect novel attacks.

Kushwaha and Ranga [3] proposed SaE-ELM-Ca. Although this method was designed to inevitably determine the appropriate hidden neurons number to improvise model's learning capability, this method failed to utilize multiple connections for testing and instead used single connection. This drawback was hopefully eradicated in [1]. Alduailij et al. [1] proposed MI and RFF, which was helpful to reduce misclassification errors by using various classifiers. However, this method failed to examine with DL-based detection and this DL-based detection was enhanced in [9]. Alqarni [9] introduced the ensemble approach for DDoS detection that limited the size of the feature and dataset producing higher performance. Here, drawbacks prevailed in its time of execution, which lasted for more time. Usage of time was limited by Cil et al. [6], where feed forward-based DNN was designed. This method attained accurate and fast results within a shorter period of time. But this method preferred the compulsory training process as a large number of packages were contained in the dataset, which was not preferred in other existing approaches. Bovenzi et al. [16] implemented the MultiModal Deep AutoEncoder (M2-DAE) model for identifying the intrusions in IoT. This approach was fitted for privacy-preserving and distributed methods with high efficiency and flexibility. However, the attack classes were not evaluated in this approach. Guarino et al. [17] implemented a machine learning approach for classifying the attacks in the network. Here, an advanced set of features were considered for the early classification. This approach obtained high F-measure, but more datasets were not considered.

The review on existing methods is shown in Table 1.

2.2. Challenges. Some challenges confronted by the predominant DDoS attack in cloud computing techniques are described as follows:

(i) Probable challenge in [10] is providing proper balancing among usages of the LUCID resource including preprocessing and traffic collection, with detection accuracy that means to ensure the obligatory level of protection against DDoS attacks without making delays to services.

(ii) In method [1], MI feature selection only was utilized as this required much time with increased data dimensions for detecting an attack, whereas other feature selection techniques, such as wrapper and sequential feature selection, were not adopted for detecting DDoS and various other attacks.

(iii) The ensemble approach in [9] utilized decision trees, naive Bayes, K-NN, and SVMs as base classifiers for detecting DDoS in cloud computing with high accuracy; however, other classifiers used in this method, performed less in detection.

(iv) CIC-DDoS2019 dataset used in [6] was converted into dual various formats for efficient classification and detection of DDoS, but this method had a challenge in detecting real-time DDoS attacks and failed to check recording network traffic from IoT and VMs.

(v) Cloud computing services are usually used as a private or public data forum depending on request by humans, and its increased utilization led to various security concerns. Informative data in cloud comes under problematic threat due to network hackers, and still, it is a challenging task to detect attacks because unauthorized users can also access cloud systems, which is a weakest point of security.

3. Cloud Model

Services of cloud computing [2] have a vast number of resource pool for data mining services and allow millions of users to store, modify, and edit data. Cloud computing exhibits environment for storage of more amount of data. The cloud model consists of two important devices, known as VM and PM. The control environment in cloud computing is considered as the cloud server. Moreover, the cloud model has the resource scheduler and allocator for resource allocations. Based on request of a user, the resource scheduler assigns available resources for processing data. PM controls multiple VM operations, and VM computes devices for storing and processing data. Scheduler controls various requests and connections by providing resources consequently in an orderly manner. The DDoS attack defence system is directly linked to the resource scheduler, as this monitor presence of behaviour of anomaly in the system in a continuous manner. While the request of the user happens inside system, then the defence strategy checks the network of traffic and announces sensible request or delivers it as an attack. When this defence strategy finds the DDoS attack, then this notifies cloud server directly.

4. Developed GHLBO-Enabled DSA for DDoS Attack Detection

DDoS attacks are most serious issue among security in the network and cause risks in the cloud computing environment. Goal of this research is finding DDoS in cloud computing based on DL. Initially, simulation on cloud is carried out, and it creates a log file, which has abrupt information and this information is directed for further feature fusion. This feature fusion is carried out using DMN

Table 1: Review on existing methods.

References	Methods	Advantages	Disadvantages
Velliangiri et al. [2]	TEHO-DBN	(i) It identified the attack at earlier stages itself	(i) It required more computational time
Arul and Punidha [4]	SD-LVQ	(i) It reduced the detection strategies of the DDoS-encrypted cross-site attack	(i) Difficult to process a large amount of data
Doriguzzi-Corin et al. [10]	LUCID	(i) Less overhead processing and minimal time of detection	(i) Time of convergence and accuracy were low
Agarwal et al. [15]	FS-WOA	(i) It avoided DoS attack entry in a big-scale industry	(i) This method lacks in generating individual instantiations to detect novel attacks
Kushwaha and Ranga [3]	SaE-ELM-Ca	(i) It determined appropriate hidden neurons number to improvise model's learning capability	(i) It failed to utilize multiple connections for testing
Alduailij et al. [1]	MI and RFF	(i) It reduced miss classification error	(i) It failed to examine with DL-based detection
Alqarni [9]	Ensemble approach	(i) Limited the size of the feature and dataset producing higher performance	(i) Time of execution was high
Cil et al. [6]	Feed forward-based DNN	(i) It attained accurate and fast results within a shorter period of time	(i) It preferred the compulsory training process as the large number of packages was contained in the dataset
Bovenzi et al. [16]	M2-DAE	(i) It had high efficiency and flexibility	(i) The attack classes were not evaluated
Guarino et al. [17]	Machine learning	(i) It obtained high F-measure	(i) More datasets were not considered

[18] with the overlap coefficient. After the process of fusion of features, the data are augmented by oversampling. Next to data augmentation, DDoS attack detection is carried out using DSA [19], which is trained using the proposed optimization algorithm, named GHLBO. The GHLBO will be designed newly by integrating the gradient descent [20] and HLBO algorithm [21]. Figure 1 shows the block diagram for the proposed GHLBO enabled feature fusion for DDOS attack detection in cloud computing.

4.1. Log File Creation. The initial phase of designed DDoS detection on attacks is creating a log file that is indicated as A. Users of the cloud system access to the model of cloud via the allocator or resource scheduler. The resource allocation model consists of data regarding free devices that allocates the device to a user based on their necessities. The resource scheduler identifies every information on the log file of each user to generate A. Abrupt information is available in the log file that is unable to be directly utilized for training [2]. The log file contains the IP address and its log information that is considered as features. The original data size obtained from datasets like BOT-IoT is in the size of 100000×48 and NSL-KDD is of 10000×42. The representation of the log file with features is given as

$$A = \{f_1, f_2, \ldots, f_n\}, \tag{1}$$

where f_1 and f_2 represent the features in the log file and n represents the complete account of features.

4.2. Feature Fusion Based on DMN with the Overlap Coefficient. After the construction of A, the next step is feature fusion based on DMN [18] with the overlap coefficient. Features that are taken from the log file are fused before the detection of DDoS as this may lead to identification of attack easily.

4.2.1. Arranging Features Based on Overlap Coefficient. Features are to be arranged based on the relativeness of their closeness character for making data in the readable format and for optimizing the rate of detection. The arrangement of features is carried out by the overlap coefficient that arranges features based on their measured closeness features. The overlap coefficient is represented as

$$O_c(f_1, f_3) = \frac{|f_1 \cap f_3|}{\min\left(|f_1||f_3|\right)}, \tag{2}$$

where f_1 and f_3 represent features with the same closeness character.

4.2.2. Fusion. After the arrangement of features according to the same measured closeness, they are fused so that independent features are converted to a unique feature in order to process easily. Fusion formula is expressed as

$$F = \sum_{a=1}^{b} \frac{\partial}{d} p_a, \tag{3}$$

where F denotes the fusion of features expressed in the vector form, b is the maximum feature range, and d indicates the full feature account. Furthermore, the generation of F is carried out based on the following formula for a as

$$a = b - \frac{K}{d}, \tag{4}$$

where d is the first obtained based on K and t, which is formulated as

$$d = \frac{K}{t}; \quad 1 \le s \le t, \tag{5}$$

where K is for full amount of features and t indicates the features selected. Here, the feature size is changed to $F_{o\times k}$ from the initial size $F_{o\times b}$.

4.2.3. Generating ∂ Using the Deep Maxout Network. The fractional coefficient ∂ is generated for finding the feature fusion depending on the overlap coefficient and data records. DMN is trained to find the fractional coefficient and the architecture of DMN is explained as follows:

*(1) Architecture of DMN.*DMN [18] is one of the neural network's types, which has many numbers of layers that create hidden activations via the maxout function. Here, functions on activation are exemplified by the n^{th} layer, where hidden units are characterized to various disjunct groups. In DMN, the activation function is replaced by MMN weights and maxout units. Maxout is a common category of ReLU which achieves the maximum operation on altered linear representations. The maxout unit-based result [22] is formulated as

$$C_z(E_\nabla) = \max_{e\in[1,m]} \mathfrak{T}_{ze}, \tag{6}$$

where $\mathfrak{T}_{ze} = E_\nabla^A \mathrm{B}_{\ldots.ze} + \mathrm{GG}_{ze}$ is the parameter that is trained and m is the total number of units of subhidden linear terms.

Feature maps are formed by layering conv filters along the MMNs activation function above the local patch, and this is fed into further higher layers. Here, every hidden neuron is the maxout unit, which is denoted as multilayer generalization guarding maxout behavior, while improving construction capability of various distributions of latent ideas. This MMN is a kind of a activation function for training. Assuming input as E_∇, which is the hidden layer raw input vector, activation function is expressed as follows:

$$\begin{aligned}
\mathrm{X}_{z,e}^1 &= \max_{e\in\left[1,m_1\right]} E_\nabla^A \mathrm{B}_{\ldots..ze} + Gg_{ze},\\
\mathrm{X}_{z,e}^2 &= \max_{e\in\left[1,m_2\right]} \mathrm{X}_{z,e}^{1^A} \mathrm{B}_{\ldots..ze} + Gg_{ze},\\
\mathrm{X}_{z,e}^h &= \max_{e\in\left[1,m_i\right]} H_{z,e}^{i-1^A} \mathrm{B}_{\ldots..ze} + Gg_{ze},\\
\mathrm{X}_{z,e}^j &= \max_{e\in\left[1,m_j\right]} \mathrm{X}_{z,e}^{j-1^A} \mathrm{B}_{\ldots..ze} + Gg_{ze},\\
\mathfrak{R}_\nabla &= \max_{e\in\left[1,m_j\right]} \mathrm{X}_{z,e}^j,
\end{aligned} \tag{7}$$

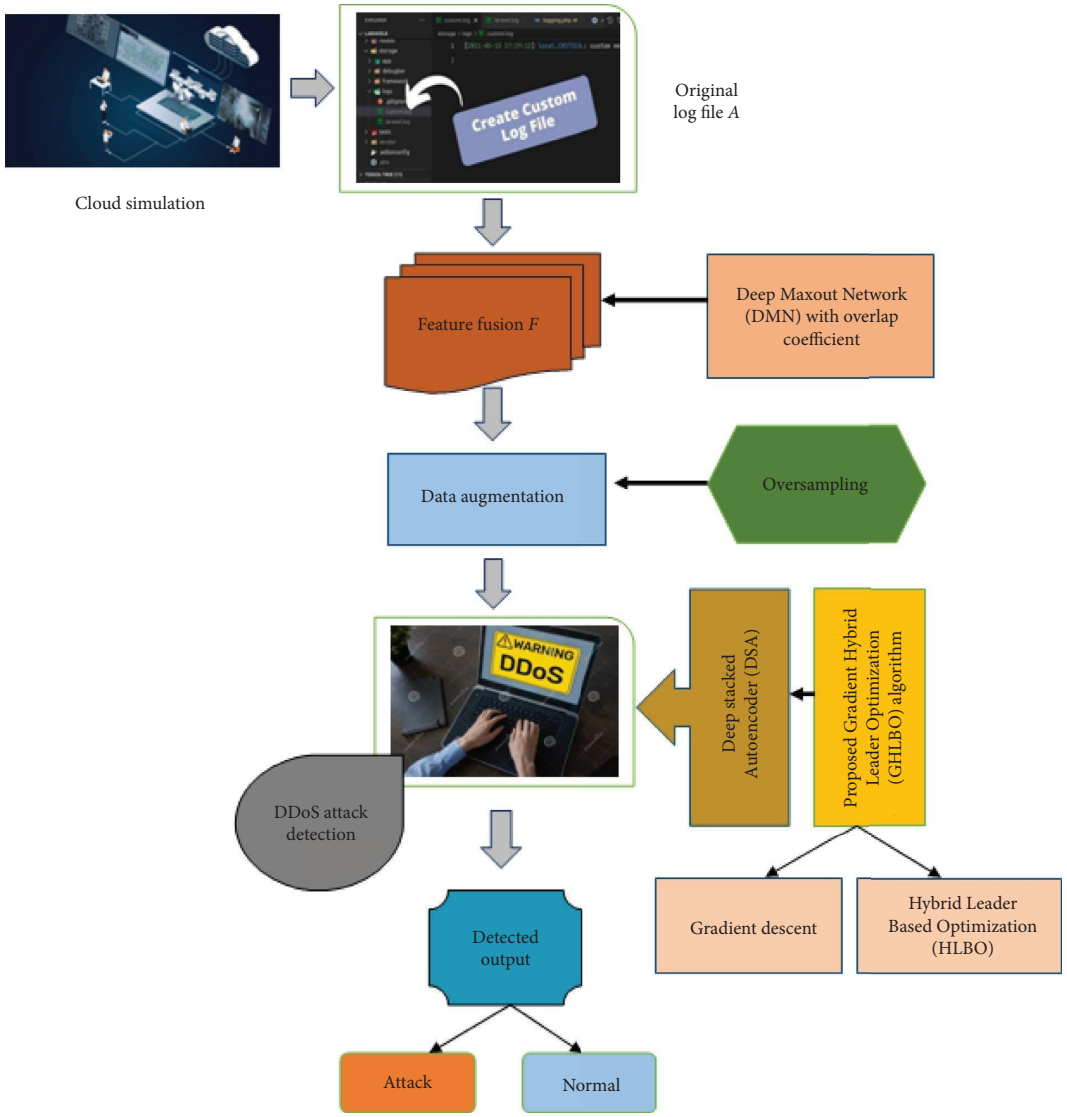

Figure 1: Block diagram for the proposed GHLBO enabled feature fusion for DDoS detection on the attack in cloud computing.

where m_i is the overall number of units in the i^{th} layer and j signifies the overall number of layers in MMN. Conventional activation functions that are nonlinear, such as the absolute value rectifier and ReLU are well approximated using MMN. Thus, feature fusion F is undergone by DMN training, from which the fractional coefficient is obtained based on the overlap coefficient that is indicated as

$$\partial = O_c(d_r, \chi_r), \tag{8}$$

where ∂ is the fractional coefficient, O_c is the overlap coefficient, d_r is the data record, and χ_r is the average of d_r belonging to the class.

After feature fusion, the size of features varies from $F_{o\times b}$ to $F_{o\times k}$. The fused features are of sizes from BOT-IoT 100000×41 to NSL-KDD 100000×31. Figure 2 represents the architecture of DMN.

4.3. Data Augmentation. Fused features F are augmented for increasing data diversity by excluding uneven balance of datasets. For eliminating imbalanced number of data, the dimensionality of the database is increased by the augmentation process. This data augmentation process is carried out using the oversampling technique. Here, the size of fused data with $(o \times b)$ is incremented to $(o \times q)$. For example, if the size of data after fusion is (10×5), then the size of data after augmentation is $(10,000 \times 5)$ that generates 99, 990 samples based on the oversampling method. Here, the augmented data is indicated as F_{aug} with size $(o \times q)$. The augmented data are

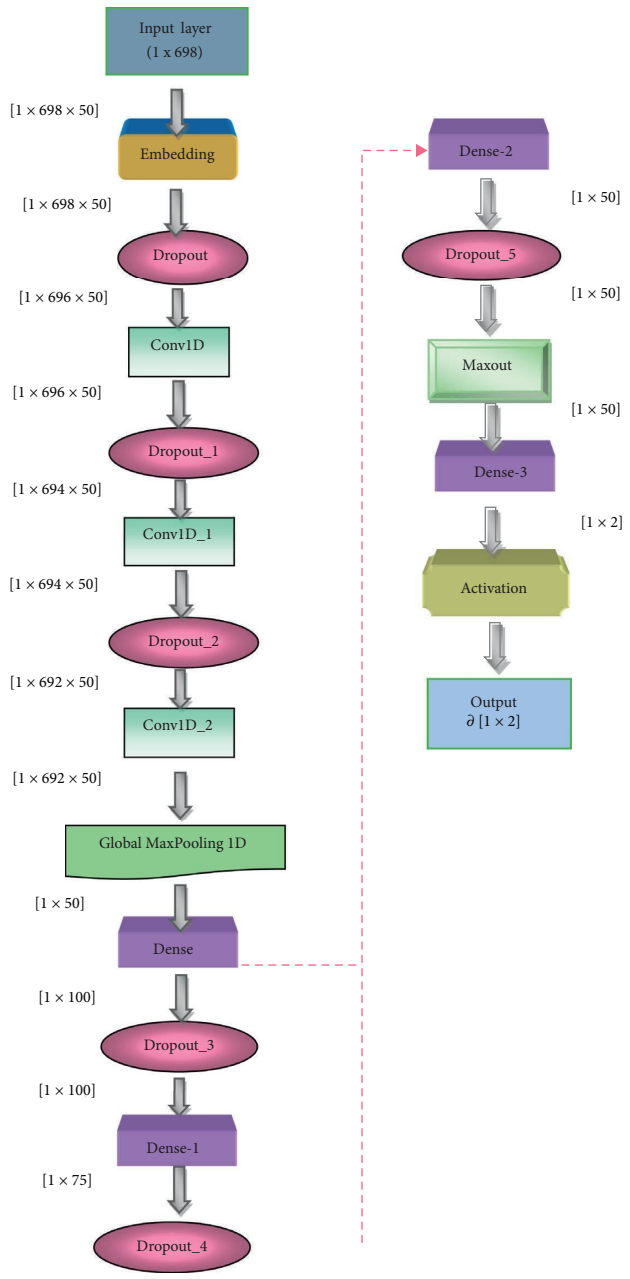

Figure 2: Structural architecture of DMN.

with sizes of 1000000×41 from the BOT-IoT dataset and 1000000×31 from the NSL-KDD dataset.

4.4. DDOS Attack Detection. After the process of data augmentation, the augmented data is fed to the next process of attack detection, where the DDoS attack is detected by DSA [19] that is trained by the proposed GHLBO. The architecture of DSA followed and the training procedure carried out is explained below.

4.4.1. Architecture of the Deep Stacked Autoencoder. An auto encoder [19] is an unsupervised learning configuration-based type, where three layers such as hidden, input, and output layers are present. The input fed to DSA is F_{aug}. Here, the training process is carried out by two sections such as the encoder and decoder. An encoder utilizes input data mapping to convert into the hidden illustration and a decoder reconstructs input data from the derived hidden illustration. For the presented unlabeled input data, $\{l_\Delta\}_{\Delta=1}^{D}$, where $l_\Delta \in Q^{I\times J}$, α_Δ indicates the vector of the hidden encoder taken from β_Δ and $\wedge$the vector of the output layer decoder is represented by l_Δ . Thus, the encoding process is formulated by

$$\beta_\Delta = \alpha\left(\mathrm{E}_1 l_\Delta + \mathrm{H}_1\right), \tag{9}$$

where the function of encoding is indicated by α, the matrix of encoder weight is E_1, and H_1 is the bias vector. The decoder process is stated by

$$\hat{l}_\Delta = P\left(\mathrm{E}_2\beta_\Delta + \mathrm{H}_2\right), \tag{10}$$

where the function of decoding is represented using P, the weight matrix of the decoder is E_2, and the bias vector is given as H_2.

For minimization of the reconstruction error, an autoencoder parameter set is optimized as

$$\varepsilon(\mathrm{O}) = \underset{\varphi,\varphi'}{\operatorname{argmin}} \frac{1}{\Delta}\sum_{r=1}^{\Delta} M\left(l^{\wedge}, \hat{l}^{r}\right), \tag{11}$$

where M is the loss function $M(l, \hat{l}) = ||l - \hat{l}||^2$.

Hence, SAE is carried out using three steps. First, the input data trains an autoencoder and thus attains the learned feature vector. Second, input for the following layer is taken as the previous layer's feature vector and this iteration is continued until training completion. Finally, hidden layer training is carried out and the backpropagation method is used for minimization of the cost function and weights are updated by the labelled tuning group for obtaining best training. Hence, output obtained from DSA is Z_d. Figure 3 exhibits structural architecture of DSA with 90% of training data.

4.4.2. Training of DSA Using Developed GHLBO. Training of DSA [19] is carried out by the developed GHLBO algorithm for the detection of DDoS attacks. GHLBO is formed by integration of the gradient descent [20] and HLBO algorithm [21]. Gradient descent is one of the most famous algorithms that perform optimization of neural networks. Various behaviours of algorithms tend to optimize this gradient descent for brief summarization to resolve challenges in those algorithms. HLBO is an optimization algorithm introduced to guide population under hybrid leader guidance where this leader is generated depending on three members, such as one random member, the next corresponding member, and the last best member. HLBO is followed by two stages, namely, exploitation and exploration. Here, each member in population is a searcher to solve issues corresponding to the space search and hence the global search forms the main criterion in HLBO. The feature of gradient descent for exaggerating the optimization

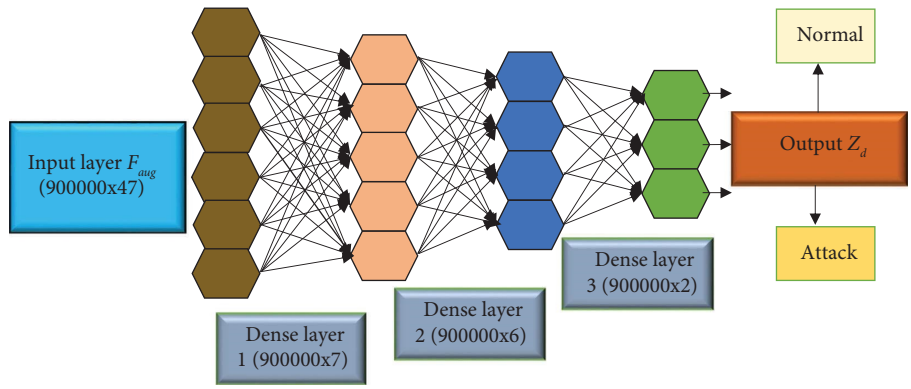

Figure 3: Structural architecture of DSA.

features of algorithms tends HLBO to more prominently improve its performance for enhancing the detection rate of DDoS attacks using newly developed and integrated GHLBO. The procedure regarding attack detection is given as follows.

(1) Initialization. In HLBO, every member in population is a searcher in threat eradicating space, and hence, all followers in population are able to enhance their own position for finding the best solution. The updating process of population is carried out based on the best member and worst member prevent algorithm from the global search in the problem eradicating space. Consider HLBO population modelled in the matrix form as

$$L = \begin{bmatrix} L_1 \\ \vdots \\ L_u \\ \vdots \\ L_Y \end{bmatrix}_{Y\times v} = \begin{bmatrix} l_{11}\cdots l_{1y}\cdots l_{1v} \\ \vdots \ddots \vdots \ddots \vdots \\ l_{u1}\cdots l_{uy}\cdots l_{uv} \\ \vdots \ddots \vdots \ddots \vdots \\ l_{Y1}\cdots l_{Yy}\cdots l_{Yv} \end{bmatrix}_{Y\times v}, \tag{12}$$

where the HLBO algorithm is denoted as L, L_u is the u^{th} candidate solution, l_{uy} is the y^{th} variable determined by the u^{th} candidate solution, Y is the HLBO population size, and v is the count of problem variables.

(2) Fitness Computation. Better optimal solution is generated by computing the fitness factor and is expressed as

$$\text{Fitness} = \frac{1}{\kappa}\sum_{\varpi=1}^{\kappa}\left[D_o - Z_d\right], \tag{13}$$

where D_o is the output aimed, the DSA output result is represented by Z_d, κ is the number of training samples taken for the training process, and ϖ is the number of processed samples.

(3) Exploration Stage. Exploration is a feature, enabling members of the population to scan various sources of the search space for finding the original optimal area. The best member in the population reduces time for exploration of the search space; however, the hybrid leader tends to update the position of members in the population. In constructing the random leader, three populations, such as random, corresponding, and best members are considered. Quality of each population member is represented as

$$w_u = \frac{T_u - T_{\text{worst}}}{\sum_{y=1}^{Y}\left(T_y - T_{\text{worst}}\right)}, u \in \{1, 2, \cdots, Y\}. \tag{14}$$

Participation coefficients of each member are expressed based on equation (14) as

$$\text{WV}_u = \frac{w_u}{w_u + w_{\text{best}} + w_\vartheta}, \tag{15}$$

$$\text{WV}_{\text{best}} = \frac{w_{\text{best}}}{w_u + w_{\text{best}} + w_\vartheta}, \tag{16}$$

$$\text{WV}_\vartheta = \frac{w_\vartheta}{w_u + w_{\text{best}} + w_\vartheta}, \tag{17}$$

where $u, \vartheta \in \{1, 2, \ldots, Y\}, \vartheta \neq u, w_u$ is the quality of theu^{th} candidate solution, w_{best} is the objective function of the best solution, and T_{worst} is the objective worst candidate function's solution. At each repetition, the hybrid leader is created for each member of the population that is represented as

$$\text{MX}_u = \text{WV}_u.\, L_u + \text{WV}_{\text{best}}.\, L_{\text{best}} + \text{WV}_\vartheta.\, L_\vartheta, \tag{18}$$

where MX_u is the hybrid leader for the u^{th} member and L_ϑ is the population member selected randomly.

(4) Updating Position. The position is updated by the hybrid leader for the optimal search space and this update position is only accepted when the objective function value is improved from the previous position. This update condition is expressed as

$$l_{u,y}(S+1) = l_{u,y}(S) + U.\left(\text{MX}_{u,y} + \text{Z}.\, l_{u,y}\right), \text{when } T_{\text{MX}_u} < T_u, \tag{19}$$

$$l_{u,y}(S+1) = l_{u,y}(S) + U.\,\text{MX}_{u,y} + U.\,\text{Z}.\, l_{u,y}, \tag{20}$$

$$l_{u,y}(S+1) = l_{u,y}(S)[1 + U.\,\mathrm{Z}] + U.\,\mathrm{MX}_{u,y}, \tag{21}$$

where $l_{u,y}(S+1)$ is the position of the u^{th} solution in the y^{th} dimension at the iteration $(S+1)$, U is a randomly formed real number from the interval (0, 1), then Z is an integer randomly selected with the set {1, 2}, and $\mathrm{MX}_{u,y}$ indicates the hybrid leader of the u^{th} solution.

(5) Updating Parameter for the Training Sample. Gradient descent updates a parameter to every training data for improving its performance and is formulated as

$$l_{u,y}(S+1) = l_{u,y}(S) - \hbar\ell f\left(l_{u,y}(S)\right), \tag{22}$$

$$l_{u,y}(S) = \hbar\ell f\left(l_{u,y}(S)\right) + l_{u,y}(S+1). \tag{23}$$

By substituting equation (23) in equation (21),

$$l_{u,y}(S+1) = \left[\hbar\ell f\left(l_{u,y}(S)\right) + l_{u,y}(S+1)\right](1 + U.\,\mathrm{Z}) + U.\,\mathrm{MX}_{u,y}, \tag{24}$$

$$l_{u,y}(S+1) - l_{u,y}(S+1)(1 + U.\,\mathrm{Z}) = \hbar\ell f\left(l_{u,y}(S)\right)(1 + U.\,\mathrm{Z}) + U.\,\mathrm{MX}_{u,y}, \tag{25}$$

$$l_{u,y}(S+1)[1 - 1 - U.\mathrm{Z}] = \hbar\ell f\left(l_{u,y}(S)\right)(1 + U.\mathrm{Z}) + U.\mathrm{MX}_{u,y}, \tag{26}$$

$$l_{u,y}(S+1) = \frac{1}{-U.\mathrm{Z}}\left[\hbar\ell f\left(l_{u,y}(S)\right)(1 + U.\mathrm{Z}) + U.\mathrm{MX}_{u,y}\right], \tag{27}$$

$$l_{u,y}(S+1) = \frac{1}{U.\mathrm{Z}}\left[\hbar\ell f\left(l_{u,y}(S)\right)(-1 - U.\mathrm{Z}) - U.\mathrm{MX}_{u,y}\right], \tag{28}$$

where the position of the y^{th} dimension in the u^{th} solution is $l_{u,y}(S+1)$, iteration is $(S+1)$, the randomly created real number is U from the interval (0, 1), the integer that is randomly selected is Z within the set {1, 2}, the hybrid leader of the u^{th} solution is $\mathrm{MX}_{u,y}$, and $\hbar$ is a parameter, which scales the gradient.

(6) Exploitation Stage. Ability to make the algorithm population enable for searching locally is termed as the exploitation phase. This brings out the best solution nearby obtained solutions. This is created by the neighbourhood member around each and every member of the population that makes the particular member to change the position and supports to find the best value for the objective-based function. Equation for exploration which is expressed as

$$l_{u,y}(S+1) = l_{u,y}(S) + (1 - 2U).\zeta\left(1 - \frac{S}{R}\right).l_{u,y}, \tag{29}$$

$$L_u = \begin{cases} L_u(S+1), T_u(S+1) < T_u, \\ L_u \quad \text{else}, \end{cases} \tag{30}$$

where ζ is the constant value equal to 0.2, $L_u(S+1)$ is the newly formed position of the u^{th} member, $l_{u,y}(S+1)$ is its j^{th} dimension, $T_u(S+1)$ is the objective function depending on the exploitation phase, S denotes the iteration counter, and R is maximum iteration numbers.

(7) Repetition. The iteration process is continued by implementing exploration and exploitation phases. The algorithm follows the next iteration stage and the process is updated and continued based on the exploration and exploitation phases. Finally, the best member solution is formed as the solution to issue.

(8) End. Till obtaining the proper optimal solution, the process gets repeated to find DDoS detection on an attack in cloud computing. Table 2 predicts explanative pseudocode of the GHLBO algorithm.

Hence, the developed GHLBO-based DSA is very efficient in DDoS attack detection in cloud computing to find whether attacked or not.

5. Discussion with Results

Results regarding DDoS attack detection depending on evaluation metrics are deliberated in this section.

5.1. Experimental Assessment. The developed model is setup in the MATLAB tool in a PC with the Intel i3 core processor, along with Windows 10 OS and 2 GB RAM.

5.2. Dataset Description. Input data for the processing of DDoS attack detection is taken from a dataset [23, 24] that has various data corresponding to attack detection.

5.2.1. NSL-KDD. NSL-KDD is updated sort of KDD cup99 that forms an efficient benchmark for researchers to compare various types of the IDS dataset. They provide 21 predicated labels with fifty thousand information. They have superfluous records in the train set with best detection rates on all frequently used records. Simultaneously, evaluation

Table 2: Proposed GHLBO's pseudocode.

Initiate GHLBO
Input: l
Adjust Y and R
Start with member position and evaluating objective function
For $u = 1$ to Y
For $S = 1$ to R
Computation of fitness using equation (13)
Stage 1: exploration
Calculation of quality by equation (14)
Calculation of participation coefficients by equations (15)–(17)
Creating hybrid leader by equation (18)
Calculating new position of u^{th} member by equation (19)
Updating gradient parameter for training sample by equation (22)
New position of u^{th} solution in y^{th} dimension is obtained by equation (28)
Stage 2: exploration
Calculation of novel position of u^{th} member using equation (29)
Updating u^{th} member by equation (30)
End if;
Recalculating best optimal solution using equation (13)
Concluded
Outcome: best member solution is generated
End GHLBO

results of various research works are provided, that is, consistent and comparable.

5.2.2. BOT-IoT. The BoT-IoT dataset was generated to design accurate environment of the network in Cyber Range Lab of Center of UNSW Canberra Cyber. The source file is provided in various formats, such as csv files, original pcap files, and argus files. These files are parted, depending on the category and subcategory of attacks, to support the process of labelling. Captured pcap files are of 69.3 GB size, with more than 72,000,000 record files.

5.3. Assessing with Performance Metrics. Performance measures utilized in this developed model is TPR, TNR, and testing accuracy. Metrics used are described as follows:

(a) TPR: TPR determines the proportion of the DDoS attack that is identified appropriately from the original file. It is indicated by using the following formula:

$$\text{TPR} = \frac{t_{\text{pr}}}{t_{\text{pr}} + f_{nr}}. \tag{31}$$

(b) TNR: this gives ratio of authentic data identified approximately from the overall number of data that is classified as true or reliable and is presented as

$$\text{TNR} = \frac{t_{nr}}{t_{nr} + f_{\text{pr}}}. \tag{32}$$

(c) Testing accuracy: it is most important measure for finding effectiveness of the developed DDoS detection approach. This gives the overall proportion of correctly identified data either attack or normal from total count of data provided and is formulated as

$$\text{Acc} = \frac{t_{\text{pr}} + t_{nr}}{t_{\text{pr}} + t_{nr} + f_{\text{pr}} + f_{nr}}. \tag{33}$$

Here, t_{pr} indicates the number of manipulated images that are found, t_{nr} is the number of authentic data, f_{pr} indicates the number of authentic data categorized as fake and f_{nr} specifies the total forged data detected as reliable.

5.4. Algorithmic Assessment. The proposed GHLBO-enabled DSA is assessed algorithmically in comparison with various other optimization techniques, such as GA [25] enabled DSA, PSO algorithm [26] enabled DSA, CS algorithm [27] enabled DSA, and HLBO enabled DSA with varying learning data in percentage. Here, the DSA is training with other optimization algorithms, such as GA, PSO, the CS algorithm, and HLBO and the performance is compared with the proposed GHLBO.

5.4.1. Algorithmic Analysis Based on BOT-IoT. BOT-IoT-based algorithmic analysis with varying percentages of learning data for various methods is discussed and represented in Figure 4. For this analysis, the learning data varies from 50% to 90% and the maximum performance is attained at 90% of learning data. Testing accuracy based the algorithmic assessment for the BOT-IoT dataset is indicated in Figure 4(a). If learning data is 50%, the testing accuracy value is 0.798 for GA + DSA, 0.779 for PSO + DSA, 0.824 for CS + DSA, 0.878 for HLBO + DSA, and 0.896 for proposed GHLBO + DSA with performance improvement of 10.957%, 13.021%, 8.049%, and 1.961%. Figure 4(b) shows the TPR-based algorithmic analysis for the BOT-IoT dataset. Here, GA + DSA shows the TPR value of 0.794, PSO + DSA shows 0.828, CS + DSA gives the value of 0.848, HLBO + DSA gives 0.869, where the proposed method attains TPR of 0.879 when learning data is 60%. The performance improvement in the TPR value with the proposed model is 9.589%, 5.773%, 3.462%, and 0.990%. The TNR variation with algorithmic analysis from the BOT-IoT dataset is indicated in Figure 4(c). If learning data percentage is 70, TNR values are 0.786, 0.816, 0.783, 0.856, and 0.865 for GA + DSA, PSO + DSA, CS + DSA, HLBO + DSA, and proposed GHLBO + DSA. The improvement in performance values of TNR is 9.081%, 5.660%, 9.443%, and 0.990%.

5.4.2. Algorithmic Analysis Based on NSL-KDD. The algorithmic assessment with change in the percentage of learning data from NSL-KDD is given in Figure 5. Testing accuracy-based analysis for the algorithm is depicted in Figure 5(a). If learning data is 80%, then the testing accuracy value for the proposed model is 0.894, whereas other methods show lesser values of 0.726 for GA + DSA, 0.834 for PSO + DSA, 0.874 for CS + DSA, and 0.891 for HLBO + DSA. The value of testing accuracy is improved with the ranges of 18.789%, 6.689%, 2.171%, and 0.310%. The TPR-based algorithmic assessment for NSL-KDD is depicted in Figure 5(b). Here,

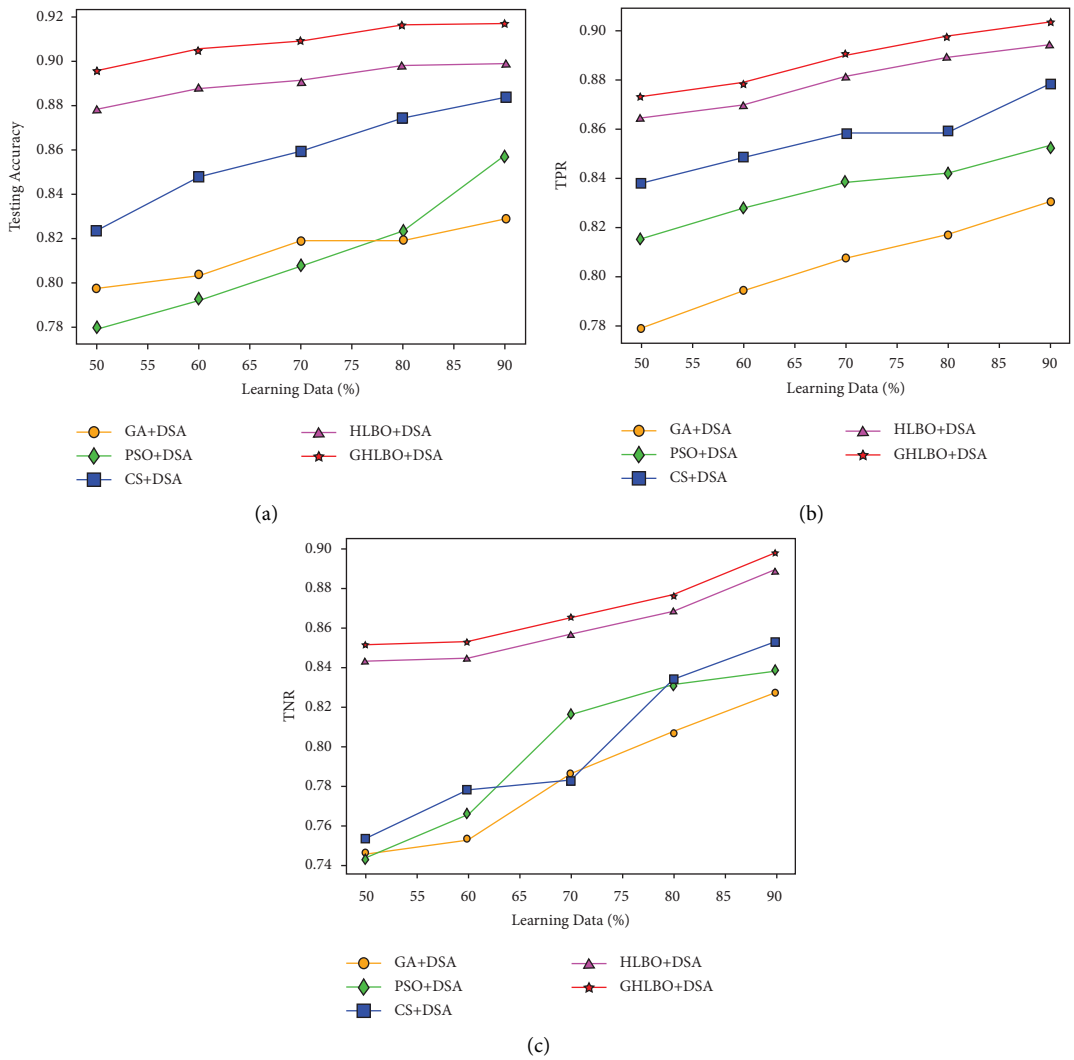

Figure 4: The algorithmic assessment based on BOT-IoT, (a) testing accuracy, (b) TPR, and (c) TNR.

when percentage of learning data = 90, TPR is 0.847 for GA + DSA and increases with values of 0.883, 0.887, 0.896, and 0.909 for PSO + DSA, CS + DSA, HLBO + DSA, and developed method. This shows improvement in performance with the proposed model with values of 6.803%, 2.905%, 2.509%, and 1.542%. Figure 5(c) gives the TNR variation of algorithmic analysis with respect to NSL-KDD. When learning data = 50%, the TNR value for the proposed method is 0.896, and it changes for PSO + DSA, GA + DSA, CS + DSA, and HLBO + DSA with values of 0.885, 0.873, 0.879, and 0.842, accordingly with performance improvement of 2.603%, 1.251%, 1.848%, and 5.982%.

5.5. Comparative Assessment. Developed model is compared with various methods, such as TEHO-DBN [28], LUCID [10], the ensemble approach [29], DNN [30], SD-LVQ [4], and FS-WOA [15] by changing learning data.

5.5.1. Comparative Analysis Based on BOT-IoT. Figure 6 depicts comparative assessment of various methods in terms of BOT-IoT. Testing accuracy based comparative analysis is indicated in Figure 6(a). When learning data percentage = 60, then values of testing accuracy are 0.788, 0.799, 0.833, 0.879, 0.883, 0.886, and 0.897 for TEHO-DBN, LUCID, the ensemble approach, DNN, SD-LVQ, FS-WOA, and the proposed method. Improvement in performance with developed model for testing accuracy is 12.159%, 10.920%, 7.184%, 1.961%, 1.56%, and 1.23%. Figure 6(b) shows the TPR-based comparative assessment in terms of BOT-IoT. For, 70% learning data, values of TPR are 0.802, 0.829, 0.844, 0.883, 0.889, 0.892, and 0.901 for TEHO-DBN, LUCID, the ensemble approach, DNN, SD-LVQ, FS-WOA, and the proposed method. This shows improvement in performance with 10.997%, 7.936%, 6.367%, 1.961%, 1.33%, and 1%. Figure 6(c) depicts TNR-based comparative analysis in terms of BOT-IoT. When learning data = 80%, TNR values of TEHO-DBN is 0.809, LUCID is 0.836, the ensemble approach is 0.841, DNN is 0.890, SD-LVQ is 0.902, FS-WOA is 0.903, and the proposed method is 0.908. Performance improvement with the developed model in terms of TNR is 10.915%, 7.889%, 7.367%, 1.961%, 0.66%,

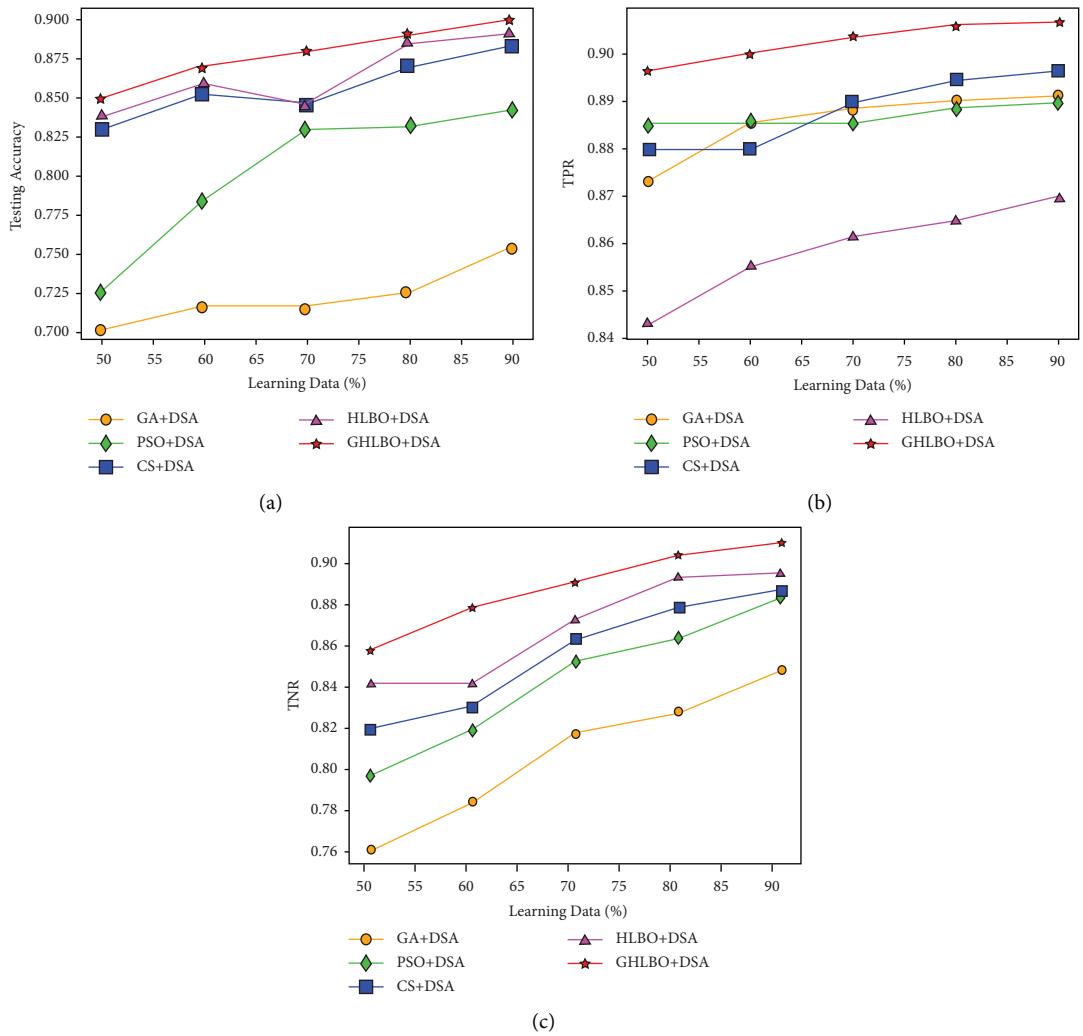

Figure 5: The algorithmic assessment based on NSL-KDD, (a) testing accuracy, (b) TPR, and (c) TNR.

and 0.55%. The ROC analysis in terms of BOT-IoT is shown in Figure 6(d). When TPR = 3, FPR value of TEHO-DBN is 0.701, LUCID is 0.817, ensemble approach is 0.832, DNN is 0.833, SD-LVQ is 0.836, FS-WOA is 0.839, and the proposed method is 0.867.

5.5.2. Comparative Analysis Based on NSL-KDD. Figure 7 depicts the comparative assessment of many methods in terms of NSL-KDD. Testing accuracy based comparative performance is depicted in Figure 7(a). When learning data = 90%, then testing accuracy values are 0.828 for TEHO-DBN, 0.848 for LUCID, 0.878 for the ensemble approach, 0.896 for DNN, 0.898 for SD-LVQ, 0.908 for FS-WOA, and 0.914 for the proposed method. Improvement in performance with the developed model for testing accuracy is 9.378%, 7.167%, 3.851%, 1.961%, 1.75%, and 0.66%, respectively. Figure 7(b) shows the TPR-based comparative assessment in terms of NSL-KDD. For, 50% learning data, values of TPR are 0.799, 0.813, 0.834, 0.869, 0.872, 0.877, and 0.887 for TEHO-DBN, LUCID, the ensemble approach, DNN, SD-LVQ, FS-WOA, and the proposed method. This shows improvement in performance with 9.895%, 8.416%, 5.954%, 1.961%, 1.69%, and 1.13%. Figure 7(c) depicts TNR-based comparative analysis in terms of NSL-KDD. When learning data = 60%, TNR values are TEHO-DBN = 0.782, LUCID = 0.771, the ensemble approach = 0.782, DNN = 0.841, SD-LVQ = 0.844, FS-WOA = 0.849, and the proposed method = 0.857. The performance improvement with the developed model in terms of TNR is 8.817%, 10.078%, 8.746%, 1.961%, 1.52%, and 0.93%. The ROC analysis in terms of NSL-KDD is shown in Figure 7(d). When TPR = 3, FPR values of TEHO-DBN is 0.880, LUCID is 0.876, the ensemble approach is 0.881, DNN is 0.853, SD-LVQ is 0.860, FS-WOA is 0.869, and the proposed method is 0.894.

5.6. Discussion with Comparison. Comparison is carried out for three evaluation metrics with respect to dual datasets, such as BOT-IoT and NSL-KDD for 90% learning data that is depicted in Table 3. For 90% learning data, data taken from the BOT-IoT dataset shows the maximum testing accuracy value of 0.917, the TPR value of 0.908, and the maximum

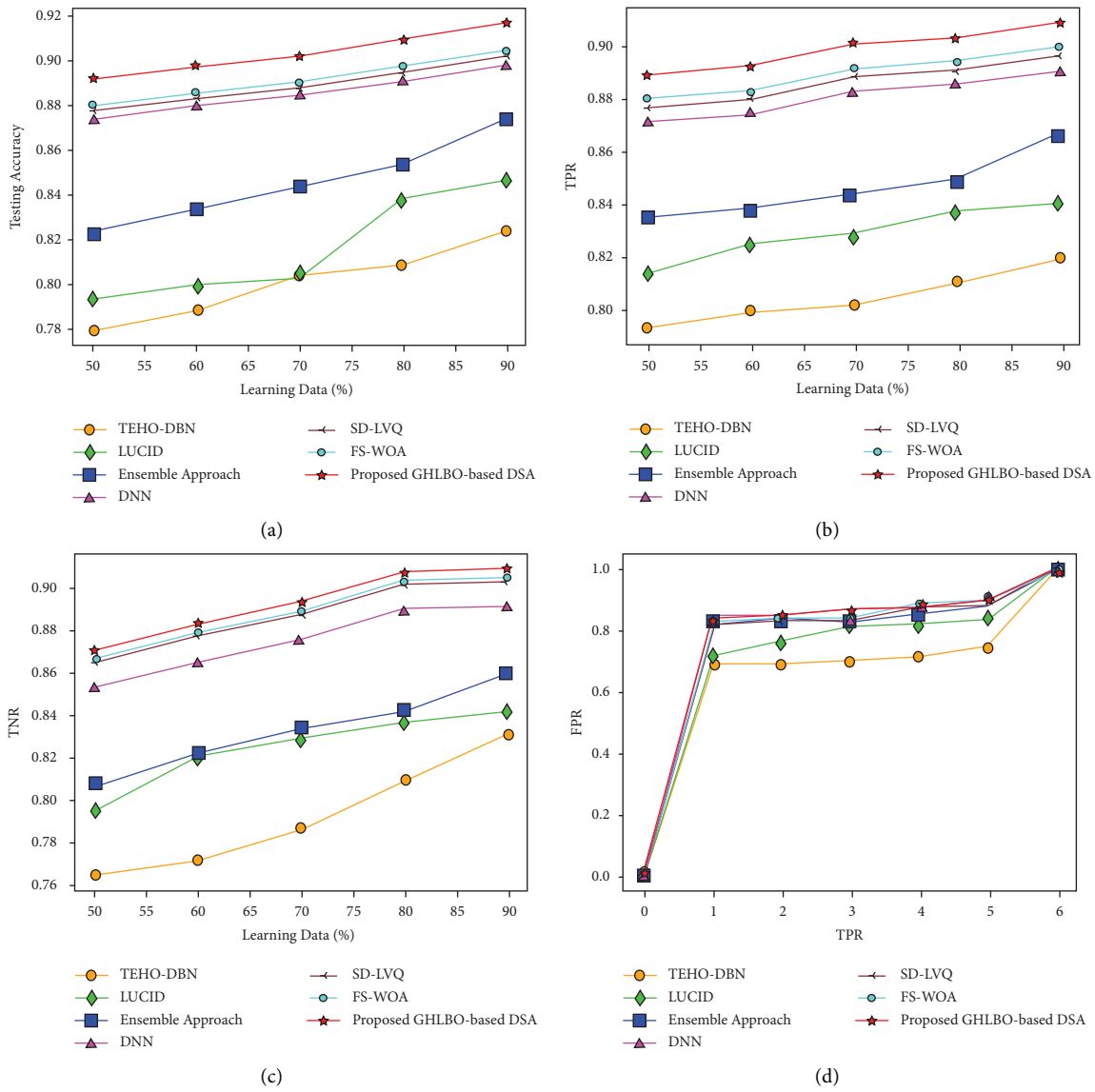

Figure 6: Comparative analysis in terms of BOT-IoT, (a) testing accuracy, (b) TPR, (c) TNR, and (d) ROC.

TNR value of 0.909. Hence, proposed GHLBO-enabled DSA is a very efficient method with high ranges of testing accuracy, TPR, and TNR, when compared with other existing methods.

Table 4 shows the computational analysis of the GHLBO-based DSA and TEHO-DBN, LUCID, the ensemble approach, DNN, SD-LVQ, and FS-WOA. The minimum computational time of the GHLBO-based DSA is 2.676 sec.

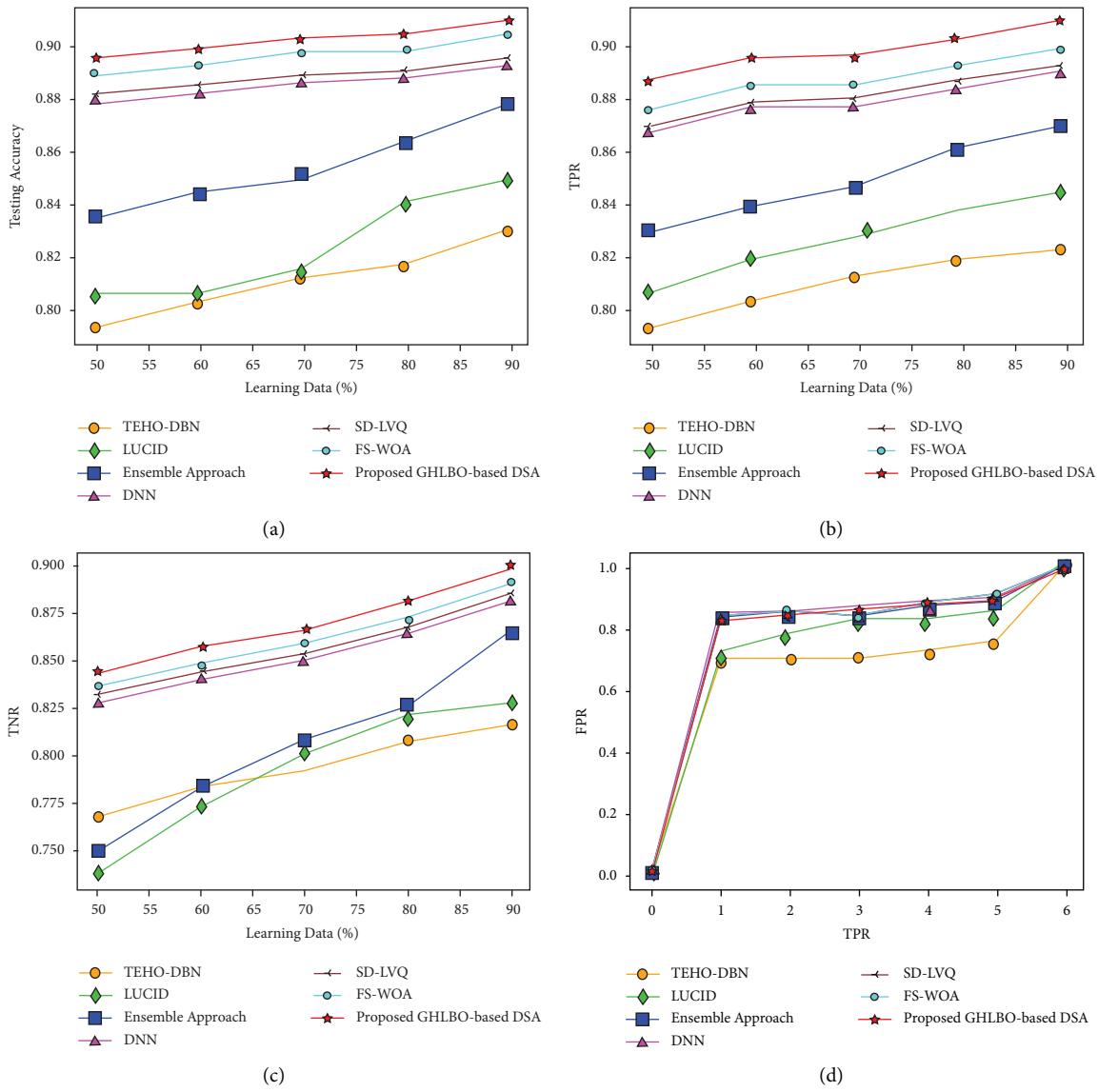

Figure 7: Comparative assessment based on NSL-KDD, (a) testing accuracy, (b) TPR, (c) TNR, and (d) ROC.

Table 3: Discussion with comparison of the proposed technique with existing techniques.

Classification types	Methods/ metrics	TEHO-DBN	LUCID	Ensemble approach	DNN	SD-LVQ	FS-WOA	Proposed GHLBO-based DSA
BOT-IoT with 90% learning data	Testing accuracy	0.824	0.846	0.873	0.899	0.902	0.905	**0.917**
	TPR	0.819	0.840	0.866	0.891	0.896	0.900	**0.909**
	TNR	0.831	0.842	0.860	0.891	0.903	0.905	**0.909**
NSL-KDD with 90% learning data	Testing accuracy	0.828	0.848	0.878	0.896	0.898	0.908	0.914
	TPR	0.828	0.848	0.871	0.891	0.894	0.899	0.909
	TNR	0.816	0.827	0.866	0.883	0.886	0.892	0.901

Bold values show higher performance compared to other methods.

Table 4: Computational time analysis.

Methods	TEHO-DBN	LUCID	Ensemble approach	DNN	SD-LVQ	FS-WOA	Proposed GHLBO-based DSA
Computational time (sec)	7.325	6.895	4.366	3.636	5.532	4.321	**2.676**

Bold values show higher performance compared to other methods.

6. Conclusion

Cloud computing transforms the IT infrastructure into utility and its characteristics such as utilising virtualisation, relying on the Internet for services, and multiple tenants inherently making the security of the network a major and unpredictable obstacle. The insider DDoS attack is a primary challenge for any cloud operational environment because it deactivates the service completely, and hence, DDoS attack should be completely eradicated as they vary the performance of cloud. In this article, DDoS attacks are detected easily in an optimized way by the proposed GHLBO algorithm. This optimized algorithm is helpful in training DSA that finds attacks in an efficient manner. Here, DMN with the overlap coefficient is responsible for the feature fusion process, and augmentation of data is carried out by oversampling technique. Also, the proposed GHLBO is generated by integrating gradient descent with the HLBO algorithm. Moreover, this proposed method is analyzed by three performance metrics such as TPR, TNR, and testing accuracy with values of 0.909, 0.909, and 0.917. However, the overhead analysis was not considered in the proposed method. This will be considered in the further extension of the devised approach. Also, the advanced optimization method will be included in this approach for better performance and more performance metrics will be considered for the performance evaluation.

Nomenclature

IT:	Information technology
DDoS:	Distributed denial of service
GHLBO:	Gradient hybrid leader optimization
DSA:	Deep stacked autoencoder
DMN:	Deep maxout network
HLBO:	Hybrid leader-based optimization
TPR:	True positive rate
TNR:	True negative rate
IaaS:	Infrastructure-as-a-service
SaaS:	Software-as-a-service
PaaS:	Platform-as-a-service
DoS:	Denial of service
DL:	Deep learning
DNN:	Deep neural network
SA:	Stacked autoencoder
DT:	Decision tree
URL:	Uniform resource locator
DBN:	Deep belief neural network
TEHO:	Taylor-elephant herd optimization
MLP:	Multilayer perceptron
SD-LVQ:	Supervised deep learning vector quantization
FS-WOA:	Feature selection-whale optimization algorithm
SaE-ELM-Ca:	Self-adaptive evolutionary extreme learning machine with crossover adaptation
MI:	Mutual information
RFF:	Random forest feature
K-NN:	k-nearest neighbour
SVM:	Support vector machine
CIC-DDoS:	Canadian Institute for Cybersecurity-DDoS
IoT:	Internet of Things
VM:	Virtual machine
PM:	Physical machine
BOT-IoT:	Robot-Internet of Things
IP:	Internet protocol
MMN:	Multimaxout network
ReLU:	Rectified linear unit
IDS:	Intrusion detection system
GA:	Genetic algorithm
PSO:	Particle swarm optimization
CS:	Cuckoo search
DNN:	Deep neural network.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Balasubramaniam S contributed to conceptualisation, investigation, data curation, formal analysis, and writing the original draft; Vijesh Joe C contributed to writing and formal analysis; Siva Kumar T A contributed to conceptualisation and project administration; Prasanth A contributed to project administration and writing; Satheesh Kumar K contributed to project administration, supervision, and writing, reviewing, and editing; Kavitha V contributed to supervision and writing, reviewing, and editing; and Rajesh Kumar Dhanaraj contributed to –the final review and verification.

References

[1] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij, and F. Malik, "Machine-learning-based DDoS attack detection using mutual information and random forest feature importance method," *Symmetry*, vol. 14, no. 6, p. 1095, 2022.

[2] S. Velliangiri, P. Karthikeyan, and V. Vinoth Kumar, "Detection of distributed denial of service attack in cloud computing using the optimization-based deep networks," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 33, no. 3, pp. 405–424, 2021.

[3] G. S. Kushwah and V. Ranga, "Optimized extreme learning machine for detecting DDoS attacks in cloud computing," *Computers and Security*, vol. 105, Article ID 102260, 2021.

[4] E. Arul and A. Punidha, "Supervised deep learning vector quantization to detect MemCached DDOS malware attack on cloud," *SN Computer Science*, vol. 2, no. 2, pp. 85–12, 2021.

[5] J. K. Seth and S. Chandra, "An effective DOS attack detection model in cloud using artificial bee colony optimization," *3D Research*, vol. 9, no. 3, pp. 44–13, 2018.

[6] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed forward based deep neural network model," *Expert Systems with Applications*, vol. 169, Article ID 114520, 2021.

[7] Q. Yan and F. R. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 52–59, 2015.

[8] Y. Mirsky, D. Tomer, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, Beijing China, October 2018.

[9] A. A. Alqarni, "Majority vote-based ensemble approach for distributed denial of service attack detection in cloud computing," *Journal of Cyber Security and Mobility*, vol. 12, pp. 265–278, 2022.

[10] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, "LUCID: a practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[11] F. Jiang, Y. Fu, B. B. Gupta et al., "Deep learning based multichannel intelligent attack detection for data security," *IEEE transactions on Sustainable Computing*, vol. 5, no. 2, pp. 204–212, 2020.

[12] A. Abeshu and N. Chilamkurti, "Deep learning: the Frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[13] A. Al-Abassi, H. Karimipour, A. Dehghantanha, and R. M. Parizi, "An ensemble deep learning-based cyber-attack detection in industrial control system," *IEEE Access*, vol. 8, pp. 83965–83973, 2020.

[14] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963–1971, 2020.

[15] A. Agarwal, M. Khari, and R. Singh, "Detection of DDOS attack using deep learning model in cloud storage application," *Wireless Personal Communications*, vol. 127, pp. 419–439, 2021.

[16] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proceedings of the GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, April 2020.

[17] I. Guarino, G. Bovenzi, D. Di Monda, G. Aceto, D. Ciuonzo, and A. Pescapé, "On the use of machine learning approaches for the early classification in network intrusion detection," in *Proceedings of the IEEE International Symposium on Measurements and Networking (M&N)*, IEEE, Padua, Italy, June 2022.

[18] W. Sun, F. Su, and L. Wang, "Improving deep neural networks with multi-layer maxout networks and a novel initialization method," *Neurocomputing*, vol. 278, pp. 34–40, 2018.

[19] G. Liu, H. Bao, and B. Han, "A stacked autoencoder-based deep neural network for achieving gearbox fault diagnosis," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5105709, 10 pages, 2018.

[20] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, https://arxiv.org/abs/1609.04747.

[21] P. Trojovsky and M. Dehghani, "Hybrid leader based optimization: a new stochastic optimization algorithm for solving optimization applications," *Scientific Reports*, vol. 12, 2022.

[22] G. Castaneda, P. Morris, and T. M. Khoshgoftaar, "Evaluation of maxout activations in deep learning across several big data domains," *Journal of Big Data*, vol. 6, no. 1, pp. 72–35, 2019.

[23] B. Ritu and R. Nagpal, "A review on kdd cup99 and nsl nsl-kdd dataset," *International Journal of Advanced Research in Computer Science*, vol. 10, 2022.

[24] J. M. Peterson, L. L. Joffrey, and M. K. Taghi, "A review and analysis of the bot-iot dataset," in *Proceedings of the 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Oxford, United Kingdom, July 2021.

[25] M. Kumar, D. Husain, N. Upreti, and D. Gupta, "Genetic algorithm: review and application," *Journal of Information and Knowledge Management*, vol. 2, no. 2, pp. 451–454, 2010.

[26] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.

[27] M. Mareli and B. Twala, "An adaptive Cuckoo search algorithm for optimisation," *Applied computing and informatics*, vol. 14, no. 2, pp. 107–115, 2018.

[28] S. Velliangiri and H. M. Pandey, "Fuzzy-Taylor-elephant herd optimization inspired Deep Belief Network for DDoS attack detection and comparison with state-of-the-arts algorithms," *Future Generation Computer Systems*, vol. 110, pp. 80–90, 2020.

[29] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020.

[30] R. Miikkulainen, J. Liang, E. Meyerson et al., "Evolving deep neural networks," *Artificial intelligence in the age of neural networks and brain computing*, vol. 32, pp. 293–312, 2019.