International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/50 Volume 11, Issue 6, November – December (2024)

RESEARCH ARTICLE

DDoS Attack Detection in Cloud Computing Using Optimized Elman Neural Network Based on Bacterial Colony Optimization and Centroid Opposition-Based Learning

S. Kalvikkarasi
PG and Research Department of Computer Science, Government Arts College (Autonomous) (Affiliated to Bharathidasan university, Trichy), Karur, Tamil Nadu, India.
✉ kalvijaya2021@gmail.com

A. Saraswathi
PG and Research Department of Computer Science, Government Arts College (Autonomous) (Affiliated to Bharathidasan university, Trichy), Karur, Tamil Nadu, India.
sarasdharam78@gmail.com

Received: 19 August 2024 / Revised: 02 December 2024 / Accepted: 14 December 2024 / Published: 30 December 2024

Abstract – Cloud computing infrastructures are particularly vulnerable to Distributed Denial of Service (DDoS) attacks due to the large-scale and dynamic nature of resources. Large data volumes are handled by cloud settings, which raises the computational cost of detection, and filtering malicious traffic from genuine traffic in such large quantities is difficult. The conventional detection techniques are insufficient. The optimized Elman Neural Network (ENN) used in this study's proposed enhanced DDoS attack detection framework combines centroid opposition-based learning (COBL) with bacterial colony optimization (BCO) called COBCO. The conventional BCO lacks population diversity and can fall into local optima due to random initialization and population update. To overcome the above issues, COBL is used for population initialization and population update to enhance population diversity and avoid local optima issues. By imitating bacterial foraging behavior, the COBCO algorithm improves the ENN's capacity to explore and exploit the solution space, increasing the network's speed of convergence and accuracy of detection. Meanwhile, COBL enhances the learning process by producing a wider range of solid candidate solutions, which offset the drawbacks of conventional opposition-based learning. Extensive simulations show that the suggested strategy outperforms traditional techniques in identifying different kinds of DDoS attacks.

Index Terms – Elman Neural Network, Bacterial Colony Optimization, Centroid Opposition-Based Learning, Hyperparameter Optimization, Convergence Rate, DDoS Attack Detection.

1. INTRODUCTION

Cloud computing is a capable technology that gives customers an easy way to access resources and services via the Internet. Utility computing is replacing desktop computing with this technology. On-demand self-service, resource pooling, wide network access, quick adaptability, and measurable services are among the key benefits offered by cloud technology. The cloud provides software, platforms, and infrastructure as a service through various delivery models, including private, public, and hybrid. For a few years, cloud services have been utilized by banks, hospitals, and educational institutions, following the lead of big cloud businesses [1]. Cloud computing has several advantages, but it also has several security flaws, dangers, and difficulties. The DDoS attack poses a dangerous risk to the accessibility of cloud services and resources [2]. In cloud computing, a DDoS attack is an intentional attempt to overload a targeted server, service, or network with excessive incoming data to disrupt its regular operations [3].

The ENN is intended to gather and store contextual data in a hidden layer. In 1990, Jeff Elman made it known. An input, a hidden, and an output layer make up its three layers. Through a series of stages, the hidden layer sends the activation values back to itself, storing context information. This makes it possible for the network to process data sequences by preserving knowledge about earlier inputs over time. ENNs are useful for capturing the many aspects of DDoS attacks because they can learn intricate non-linear correlations between input features[4]. They can identify minute changes in network traffic patterns that could be signs of an intrusion. Training complexity and high computational cost are common features of Elman networks. A substantial amount of time and

RESEARCH ARTICLE

money may be needed to determine the ideal architecture, hyperparameters, and training methods. Big networks with lots of traffic could find it difficult for Elman networks to scale. Large volumes of data processing in real-time may be difficult, which could cause delays or missed detections. In large networks or high-speed traffic settings, training and deploying Elman networks may demand a substantial amount of processing power.

Researchers and practitioners can effectively search the hyperparameter space and identify configurations that lead to optimal performance in DDoS attack detection by utilizing swarm intelligence (SI) techniques for hyperparameter optimization in ENNs. These techniques aid in overcoming the difficulties associated with laborious search and manual modifications, resulting in ENNs that function better [5]. ENNs hyperparameter optimization is determining an appropriate set of hyperparameters to maximize the network's performance in a particular task, such as DDoS attack detection. The BCO is a recently developed SI method demonstrated after the way bacteria hunt for food. BCO can effectively search high-dimensional and non-convex search spaces while swiftly examining the hyperparameter space and obtaining good solutions hyperparameter optimization can be achieved with it [6] and it efficiently adjusts the ERNs' hyperparameters for better performance utilizing BCO. However, the conventional BCO algorithm has many shortcomings such as population diversity, local optima, and low convergence rate. Hence, the present work developing a new improved BCO for optimizing the hyperparameters of ENN. The improved method called COBCO uses the COBL method for population unitizing and updating the population. An improvement on OBL called COBL focuses on making optimization algorithms perform better by taking the centroid of opposed solutions into account. By taking opposing solutions and their centroids into account, COBL adds even more variability. Searching regions distinct from the current solutions improves the search space research. Additionally, by directing the search towards the center of promising alternatives, it facilitates exploitation. The research aims to use the COBL that achieves an appropriate balance between exploitation and exploration of BCO. It accomplishes this by using the theory of centroid opposition, in which bacterial colonies travel in the opposite direction of the population's centroid to discover new locations while preserving ties to the promising ones. This study's primary objectives are to use a COBCO to accelerate learning, lower error, and maximize ENN performance to the global minimum. The significance of the new model lies in its ability to optimize the ENN's parameters to boost training efficiency, reduce network running time, and improve training speed and convergence. Create a detection system to increase the precision of differentiating between harmful and legitimate communication in cloud environments by utilizing the optimized ERNN. Research has the following contributions:

- The recommended COBCO+ENN is utilized to distinguish malicious data from incoming data.
- The new population initialization and position updating are proposed to improve the performance of BCO called COBCO to enhance the convergence, avoid local optima, and enhance the population diversity.
- The COBCO uses the COBL which offers a productive way to explore and utilize the solution space.
- COBCO is used to optimize hyperparameters of ENN to enhance the detection accuracy and convergence rate.
- The generated detection method was compared with a few benchmark detection algorithms for performance analysis.
- Four distinct DDoS attack datasets are examined to assess the efficacy and derive experimental conclusions.

The remaining sections are as follows: Section 2 covers various current research papers, while Sections 3, 4, 5, 6, and 7 explore research methodologies such as ENN, BCO, COBL, COBCO, and the suggested COBCO-ENN. The outcomes of the experiment are offered in Section 8, and the paper's conclusions are covered in Section 9.

2. RELATED WORKS

Relevant research, practical detection method development, and enhanced cloud environment security posture are all facilitated by related efforts in DDoS attack detection for cloud computing. Related works provide a basis for innovation and advancement in cloud computing DDoS attack detection, propelling improvements in methods, tools, and approaches to fortify cloud infrastructure security. The identification of current weaknesses and difficulties in DDoS attack detection that are unique to cloud computing systems is aided by related efforts. They shed light on the flaws in the detection techniques used today and suggest areas for development. R. Priyadarshini et al. (2022) [7] suggested a unique source-based DDoS defense strategy that may be utilized to counteract DDoS attacks which is deployed at the SDN using SDN to identify abnormal DDoS attacks. The planned work offers a DL-based detection approach that may block infected packets from causing more attacks and filters and forwards valid packets to the server using network traffic analysis mechanisms. The developed method is detected on network or Transport level layers and lacks in other layers. A. V. Kachavimath et al. (2021) [8] developed a method for detecting DDoS attacks that was put forth by extracting various sequence patterns from the recorded traffic using deep learning. This method has a high detection rate. The outcomes of the suggested methodology have shown that the long short-

RESEARCH ARTICLE

term memory (LSTM) strategy performs better and has higher accuracy than the multilayer perceptron (MLP) and convolutional neural network (CNN). Y. Sanjalawe et al. (2023) [9] developed a new hybrid DL mode based on hybridizing CNN with LSTM for identifying both legitimate and malicious traffic,. The suggested IDS performs better than the most advanced IDSs, according to the results. Thus, the suggested IDS satisfies the needs for high security, automatic, effective, and self-decision DDoS assault detection. However, both the above papers [8] and [9] produced high computation costs due to their random hyperparameters. T. Khempetch et al. (2021) [10] suggested an LSTM algorithm and deep neural network (DNN) for detection. According to the findings, deep learning represents an additional avenue for identifying potential future disruption-causing threats. D. Kumar et al. (2023) [11] developed a detection based on LSTM that was built to detect DDoS threats on a representative sample of network traffic packets. The DL approach known as LSTM once trained, updates itself; LSTM operates quickly and accurately even with fewer data points. LSTM takes more computation learning time to produce optimal results due to its random hyperparameters.

S. Potluri et al. (2020) [12] created a new technique for DDoS attacks, as well as methods for detecting and preventing them. The comprehensive analysis also describes the consequences of DDoS attacks on cloud platforms and the necessary protection methods for those effects. Y. Sanjalawe et al. (2023) [13] developed a new detection method-based DNN approach called FS-WOA-DNN, together with a novel feature selection-whale optimization methodology, to counteract DDoS attacks. The DNN classifier is applied to the chosen features to distinguish between normal and compromised data. To further strengthen the security of the suggested architecture, homomorphic encryption is used to protect regular data, which is then safely stored on the cloud**.** Datasets used for DDoS detection are frequently unbalanced since there are a lot more instances of regular traffic than attack occurrences. A. E. Cil et al. (2021) [14] recommended using the DNN as a DL model to identify attacks on the packet sample that was obtained from network traffic. Because the DNN comprises self-updating layers and incorporates extracting the features and classification procedures into its structure, it can activate quickly and precisely even with little data. S. Velliangiri et al. (2021) [15] developed a DL-based classifier to detect DDoS attacks. User service requests are gathered and organized into log information. To shorten the classifier's training time, a few key features are chosen from the log file and classified using the Bhattacharya distance measure. In this case, the Deep Belief Network (DBN) based on Taylor-Elephant Herd Optimization is created by adjusting Elephant Herd Optimization (EHO) using the Taylor series. However, it will take a high computation cost. A. Amjad et al. (2019) [16] created the DDoS attack and the method to stop it, reducing the server side's vulnerability. The scenario involves a DDoS attack against cloud-based websites, where millions or perhaps trillions of packets are sent, causing them to differ between hosts. Utilizing ParrotSec and other operating systems to enable the attack. S. UR Rehman et al. (2021) [17] developed to defend against real-world attacks, a unique, highly effective method called DIDDOS, utilizing a Gated Recurrent Unit (GRU). However, the GRU produced high accuracy. But computationally costly and demands a lot of computing power, particularly when evaluating massive amounts of network traffic in real-time.

D. Alghazzawi et al. (2021) [18] propose utilizing a hybrid DL model, specifically a CNN with BiLSTM, to accurately forecast attacks. Only the most relevant features were selected by rating that had the highest score in the given data set. However, the hybrid method produced high accuracy and lacked overfitting. A. V. Songa et al. (2023) [19] suggest a DDoS detection framework that uses Ensemble feature selection with RNN to address the current issue. It combines an RNN with an Ensemble of several machine-learning techniques. The framework aims to select the appropriate features using the ensemble of six ML algorithms. Using RNN, these chosen traits are then utilized to categorize network traffic as normal or attack. However, it has taken low computational time and produced low accuracy. S. Balasubramaniam et al. et al. (2023) [20] created a novel technique, the suggested gradient hybrid leader optimization (GHLBO), to detect DDoS attacks efficiently. The deep stacked autoencoder (DSA), which effectively identifies attacks, is trained by this optimized technique. In this case, oversampling is used to augment the data, and a deep max-out network (DMN) with an overlap coefficient is used to fuse the features. However, it has produced high accuracy and sometimes it fails to identify unknown traffic. G. S. Kushwah et al. (2021) [21] developed a new DDoS detection system built on top of a modified Self-adaptive evolutionary extreme learning machine (SaE-ELM) was described which was enhanced by adding two new characteristics. It can, first of all, adjust to the most appropriate crossover operator. Second, it is capable of automatically figuring out how many hidden layer neurons are needed. The model's capacity for classification is enhanced by these features. However, compared to the SaE-ELM-based system, it displays a longer training time. P. T. Dinh et al. (2021) [22] suggested approach, which consists of online and offline phases and implements a GRU, can lessen vanishing gradient issues by capturing complicated temporal dependent links in the data. Initially, the suggested plan acquires precise multivariate time series representations to mirror the typical patterns. Subsequently, input data reconstruction is done using these representations. Lastly, the reconstruction probabilities can be

RESEARCH ARTICLE

utilized to understand data in addition to identifying abnormalities. To lower error rates, the suggested method additionally adds a self-adjusting threshold. In contrast, current solutions typically employ a hard threshold to evaluate anomalies, which raises error rates. But it has a low convergence rate. S. Sumathi et al. (2022) [23] suggested a gradient descent DL approach with LSTM and autoencoders and decoders. The hyperparameters are tuned optimally by using a hybrid HHO (Harris Hawks optimization) and PSO. The findings showed that the suggested LSTM DL model performed better than all other models created in the literature, and the suggested hybrid optimization approach picks the key characteristics. DL models need a lot of processing power to train, especially those with a lot of layers and improper hyperparameters. The behavior and performance of DL are significantly influenced by hyperparameters. Hyperparameters are determined through the learning process, in deference to model parameters, which are learned during training. The accuracy, effectiveness, and generalizability of the model are all greatly impacted by their choice. Hence, the present research work focused on hyperparameters optimization for ENN to enhance the accuracy, and convergence rate and reduce the computational time.

3. ELMAN NEURAL NETWORK (ENNS)

Elman introduced the ENN, a common technique, in 1990 [24]. The ENN framework is shown in Figure 1. One kind of feedback neural network, the ENN, gains a recurrent layer based on the hidden layer, which adds a memory function and functions as a delay operator. It keeps the network stable globally and enables it to adapt to dynamic, time-varying features. The structure of an ENN model normally consists of four layers: The hidden layer receives the information from the input layer, whose neurons are usually linear, and uses an activation function to translate or amplify it. To establish a local ring structure, the connecting layer's job is to receive the output from the hidden layer and reply with data matching the preceding instance. Because the connecting layer unit has a deferred memory effect on the features included in previous data, the neural network's output is more influenced by the actual progress learning. The output layer is ultimately used to output the results. The structure of the BPNN serves as the foundation for the ENN, which automatically links the hidden layer's output to its input depending on the delay and storage functions of the context layer. Because of the sensitivity of this joining process to the neural network's historical data, this internal feedback mechanism can improve the neural network's ability to handle dynamic input. This recording of the dynamics to a kept internal state permits the scheme to adjust to time-varying features. An ENN consists of an input, a hidden, an output, and a recurrent layer. Each layer has one or more neurons that use a nonlinear function of the weighted sum of the input samples to convey data or samples. The mathematical model specification for the input layer is defined as equation (1),

$$X_{it}(k) = \sum_{i=1}^{n} X_{it}(k-1) \quad (1)$$

Here, an input $t-$ time and n neurons are denoted by X_{it}. Each neuron has the following input model defined as following equation (2),

$$net_{jt}(k) = \sum_{i=1}^{n} W_{ij} X_{it}(k-1) + \sum_{j=1}^{p} C_j r_{jt}(k) \quad (2)$$

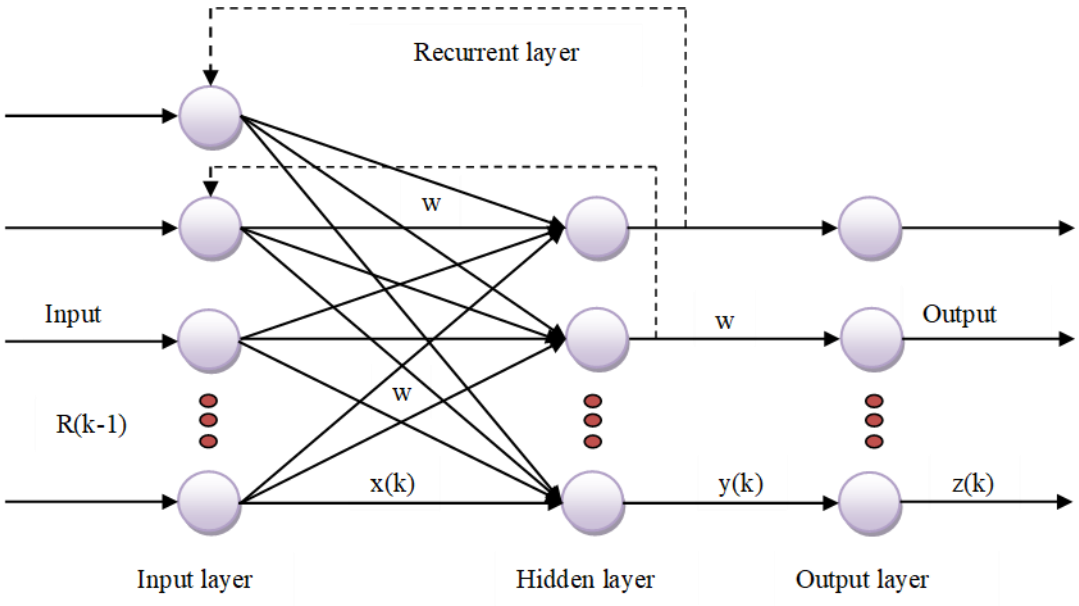

Figure 1 ENN Architecture

RESEARCH ARTICLE

W_{ij}: the hidden layer and input weights. Weights C_j are those between recurrent and hidden layers. The hidden layer is defined as follows (Equation 3):

$$Z_{jt}(k) = f(net_{jk}(k) = \sum_{i=1}^{n} W_{ij} X_{it}(k-1) + \sum_{j=1}^{p} C_j R_{jt}(k) \quad (3)$$

Equation 4 shows the definition of the recurrent layer is as follows:

$$R_{jt}(k) = Z_{jt}(k-1) \quad (4)$$

Equation 5 shows the output layer:

$$Y_t(k) = f(\sum_{j=1}^{p} V_j Z_{jt}(k) \quad (5)$$

The following equation (6) is the ENN network errors:

$$E = \sum_{k=1}^{m} (t_t - y_t)^2 \quad (6)$$

Here, t_t is the actual and y_t is the predicted value.

4. BACTERIAL COLONY OPTIMIZATION (BCO)

Niu and Wang (2012) proposed BCO, a population-based and kind of SI procedure [25] used in several practical applications [26-35]. Algorithm 1 illustrates the process of BCO. To tackle the above problem, a novel bacterial algorithm called BCO was developed with swarm intelligence characteristics to expedite the optimization process. Stages of BCO include chemotaxis and communication, elimination and reproduction, migration, and the remaining four stages. The entire BCO process takes advantage of the chemotaxis and communication phase. The bacteria use the population statistics to modify their swimming and tumbling habits. To update the positions of the microorganisms, a unique chemotaxis and communication method is applied. Throughout their lives, bacteria can be divided into two types of chemotaxis: swimming and tumbling. A stochastic direction participates in the actual swimming process when tumbling. The combined effects of the best searching director in Tumbling and the turbulent director have an impact on the updated locations and search direction of each bacterium. These effects are expressed in equation (7) as follows:

$$Pos_i(T) = Pos_i(T-1) + C(i) * [f_i.(G_{best} - Pos_i(T-1)) + (1-f_i) * (P_{best_i} - Pos_i(T-1)) + turb_i] \quad (7)$$

Bacteria lack a turbulence director, which would steer swimming toward an optimal state. This may be expressed in equation (8) as follows:

$$Pos_i(T) = Pos_i(T-1) + C(i) * [f_i.(G_{best} - Pos_i(T-1)) + (1-f_i) * (P_{best_i} - Pos_i(T-1))] \quad (8)$$

$$C(i) = C_{min} + \left(\frac{Iter_{max} - Iter_j}{Iter_{max}}\right)^n (C_{max} - C_{min}) \quad (9)$$

Where, $turb_i$- turbulent direction variance. $f_i \in \{0,1\}$. The personal and global best value is represented by P_{best} and G_{best}respectively. n is the chemotaxis step's linearly reducing method. From the equation number (9), $C(i)$ - chemotaxis step size. The maximum number of iterations and the current iteration are denoted by $Iter_{max}$ and $Iter_j$, respectively. In the phase of elimination and reproduction, the sick bacterium will be replaced by the high-energy bacterium, which will replicate them to build the newest people. The high energy shows that the bacterium hunts for resources with remarkable efficiency. The bacterium can migrate within a certain search space range during the migration phase when certain conditions are met. During the migration, the bacteria typically go toward the most recent nutrients according to a specific likelihood.

5. CENTROID OPPOSITION-BASED LEARNING (COBL)

The OBL intelligence algorithm was developed by H R Tizhoosh to obtain the opposite estimate from the current estimate and enhance the capability of the provided answer [36]. Often, population-based optimization methods begin by producing a collection of solutions. To construct the population, either past data or random selection can be applied. After that, the populations are updated using the optimization process. However, if the answer is not known in advance, the given solution is unable to converge to a global solution. Furthermore, the global solution takes longer to converge. Several studies have been carried out to mitigate these shortcomings by employing the benefits of the OBL method for population initialization and updating.

Step 1: Set up the required parameters

Step 2: For every bacterial colony

Step 3: Chemotaxis and communication

Step 4: Reproduction and elimination

Step 5: Migration

Step 6: If the final stage is not reached, step 2 should be taken; if not, the process should be terminated.

Step 7: The best position should persist in the final position

Algorithm 1 Bacterial Colony Optimization (BCO)

5.1. OBL

Given a real number x that was determined within the interval m and n let it be. The definition of the opposite number $\bar{x}$ is define in equation (10) as follows:

$$\bar{x} = m + n - x \quad (10)$$

Let $x = (x_1, x_2, \ldots, x_D)$ be a data sample with D - dimensional space. $x_1, x_2, \ldots, x_D \in R$ and $x_i[m_i, n_i] \forall i \in \{1,2,\ldots.D\}$. The opposite estimate, $\bar{x}$, is in equation (11) then defined as

RESEARCH ARTICLE

$$\bar{x}_i = m_i + n_i - x_i, i = 1,2,\ldots,D \quad (11)$$

5.2. COBL

S. Rahnamayan et al. [37] presented the COBC, an OBL scheme, and it was effectively incorporated into the DE algorithm, outperforming its competing algorithms. The total population is taken into account while calculating the centroid opposite locations in a metaheuristic method. The body's centroid can be defined as follows if x are N positions in a D - D-dimensional search space that are carrying a unit of mass which is defined in equation (12) as follows:

$$M = \frac{x_1, x_2, \ldots\ldots, x_N}{N} \quad (12)$$

The formula can be used to get the centroid point in the j^{th} dimension which is defined in equation (13) as follows,

$$M_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \quad (13)$$

Once the centroid value is known as M, the following equation (14) can be used to find the opposite point $\bar{x}_i$ of a given point x_i in the body:

$$\bar{x}_i = 2 \times M - x_i \quad (14)$$

6. COBCO METHOD

To generate the center values of the opposing values from the random numbers, the current work builds a novel COBL scheme based on center opposition values. By producing a set of centroid opposite solutions, the COBL approach increases the likelihood of obtaining higher-quality solutions. Rather than creating opposites for every possible solution, the opposites are created concerning the population's centroid. This strategy strikes a balance between exploration and exploitation by promoting searches in fresh, maybe uncharted territory while maintaining a focus on places with great potential.

Two types of modifications, including population initialization and position updating using COBL, were carried out in the BCO. In the first stage, COBL can assist in producing a variety of solutions. It is possible to generate contrary solutions by taking the centroid of the starting population into account. This increases the diversity of the starting population and may facilitate a more thorough investigation of the search space. COBL aids in striking a balance between exploitation—fine-tuning around the existing best solutions—and exploration—discovering new areas of the search space. To prevent premature convergence and guarantee that the global optimum is obtained or approximated, this balance is essential in BCO. In the second stage, the algorithm can cover the search space more effectively and possibly achieve faster convergence and higher-quality solutions by taking advantage of opposites concerning the centroid. A detailed discussion is given in the ensuing subsection.

6.1. Population Initialization Using COBL

Through population generation, the OBL algorithm enhances the provided answer. By estimating the opposite answer, $\bar{x}_i$ for x_i , an initial population, X, is created. Compute the fitness function values for both current and opposite values to get the ideal initial values for the initial population of a specific solution. Next, contrast the fitness metrics. The fitness function values are used to select the starting population, which is the new population of the optimal solution. During the initialization step, the search space range's starting population, X, is initially randomly created. Then centroid-based opposite results $\bar{X} = (\bar{x}_1, \bar{x}_2, \ldots . \bar{x}_D)$ are calculated in the dynamic search space range $[a_j, b_j]$ which are calculated as in equations (15) and (16) as follows:

$$a_j = \min_{\forall i}(x_{ij}) \quad (15)$$

$$b_j = \max_{\forall i}(x_{ij}) \quad (16)$$

Where i and j are the index of the separate result dimension correspondingly. Upon computing the centroid opposite point $\bar{x}_{ij}$, it might surpass the boundaries of the search space. The following moves it to the search space if it is greater than b_j which is defined in equation (17) as follows:

$$\bar{x}_j == a_j + \left(M_j - a_j\right) \times rand(0,1) \quad (17)$$

The fitness values are calculated for both original and opposite solutions and select the N best solutions from the set $X, \bar{X}$ according to its fitness.

6.2. Population Updating

The population update is a major endeavor that is moving the world closer to a global solution. This work uses the COBL algorithm to compute the opposite point of the present solution, which creates a new population. After the position is updated, the fitness function (MSE) values are computed for each x_i of each bacterial colony. The COBL algorithm is used to compute the opposite estimate for each position once the position has been updated, and the fitness function is then created for each $\bar{X}$. To create a new population, the best solutions are combined from the two existing populations. The previous procedure is carried out again if the stopping requirement is not satisfied. BCO incorporates a position update mechanism based on centroid opposition. The best solutions are chosen in this phase using the same methodology as the centroid opposition-based initialization phase, which involves applying COBC to the original solutions to determine the centroid opposite solutions.

7. PROPOSED COBCO-ENN

The proposed COBCO-ENN method is used to detect DDoS attacks. It can accelerate convergence and address the shortcomings of the ENN. Recently, a SI method called BCO

RESEARCH ARTICLE

was proposed. It has good accuracy and can search very wide regions for potential answers. BCO does not, however, guarantee that the best solution will be found. More specifically, because of internal iterations, the conventional BCO has a lengthy computation time and poor convergence. When the initial population and population updating are chosen inappropriately, any population-based algorithm fails to reach local optima and has a low convergence rate [38]. The suggested approach avoids the local optima problem and increases the convergence rate by combining BCO and COBL, or COBCO. Two types of adjustments are made to the CBCO to maintain a balance between exploration and exploitation: population initialization and position updating. To increase population diversity and ergodicity in the search, COBL is used once the bacteria are initiated to determine a better beginning location for each bacterium. Its fitness values are determined using the mean square error (MSE), which is in equation (18) defined as,

$$MSE = \frac{1}{N}(y_i - \hat{y}_i)^2 \quad (18)$$

where y_i - predicted value. $\hat{y}_i$ - real value. N - sample's length. The ideal weights and thresholds are represented by the data on the best bacterium. In the proposed method, normalized data is fed into the improved ENN model. ENN is trained and optimized with COBCO. Every bacterium has been chosen as a search agent to represent the original solutions. During training, each search agent's position is adjusted by minimizing the objective function (MSE). COBCO bases its search for the optimal value for ENN on the objective function and the values of the initial parameters. The output vectors are denormalized to obtain expected values. Experiments were conducted to show consistency in prediction since the SI technique can produce almost ideal solutions. Algorithm 2 and Figure 2 show the overview of the proposed COBCO-ENN detection model.

Step 1: Initialize the necessary parameters

Step 2: Initialization of the population using COBL

Step 2.1: Create a random population of (N) members, then calculate its fitness value.

Step 2.2: Determine the fitness value by computing the opposite population, $\bar{X}$

Step 2.3: Based on their fitness values, choose the best one to serve as the beginning population.

Step 3: Chemotaxis and communication

Step 4: Elimination and dispersal

Step 5: Reproduction

Step 6: Migration

Step 7: Position update using COBL

Step 7.1: The position of every bacterial colony is updated, and the fitness value of every solution is calculated.

Step 7.2: Determine the fitness value for every solution as well as the centre of opposite point for the present one.

Step 7.3: Based on the fitness function, select the appropriate solution $X \cup \bar{X}$.

Step 8: If the stopping state is met, the procedure should end; if not, move on to step 5.

Step 9: Utilize the optimal hyperparameter when training the ERNN. The best bacteria data show the optimal hyperparameters that were chosen as the hyperparameter basis. Then identify the issue in the network.

Step 10: The training is complete when it reaches the maximum number of epochs or the least error.

Algorithm 2 Proposed COBCO-ERNN

8. EXPERIMENTAL RESULTS

To obtain the optimal hyperparameters of the ENN, minimize error, and determine the optimum detection accuracy, the current work suggested novel SI-based optimization strategies termed COBCO. The optimal hyperparameters for the vectors are determined using the SI method. The COBCO algorithm uses a bacteria's position as a dimension to build the connection weights, biases value set, and learning rate that are needed by the ENN approach. The proposed COBCO-ENN technique is related to some well-known algorithms such as BCO-ENN [6], APSO-ENN [39], PSO-ENN [40], GA-ENN [41], ENN [42], BPNN [43], and SVM [44]. The related procedures are employed using MATLAB 2019b with an i5 processor and 16 GB RAM on Windows 11.

8.1. Dataset Collections

Numerous intrusion detection assessment datasets include both normal and abnormal network traffic data. The performance detection approach is analyzed using four datasets: NSL-KDD, UNSW-NB15, CIC-IDS2017, and CIC-DDoS2019. The following is a discussion of the dataset details as shown in Table 1:

a) NSL-KDD: This dataset comprises four sorts of attacks: “DOS, R2L, U2R, and Probe. KDDTrain+, KDDTest+, and KDDTest-21 have 1,25,973, 22,544, and 11,850 samples overall, respectively”. Samples from this dataset include 41 features.

b) UNSW-NB15: There are 2,540,044 samples in all in this dataset. This dataset's subset which contains 257,673 samples. There are 175,341 and 82,332 samples for training and testing, respectively. Nine different attack types are included: “analysis, backdoor, denial-of-service, exploits, fuzzers, generic, reconnaissance, shell code, and

RESEARCH ARTICLE

worms". Table 3 provides further dataset details. Samples from this dataset include 48 features [45].

Table 1 Datasets Particulars

Datasets	Attributes	Training samples	Testing samples	Total samples
NSL-KDD	41	125973	34394	160367
UNSW-NB15	48	175341	82332	257673
CICIDS2017	78	1744184	747505	2491689
CIC-DDoS2019	78	587, 966	411577	176389

c) CICIDS2017: The five days of traffic from Monday through Friday are included in this dataset. Only normal samples are present on Monday; the traffic on the other days consists of both normal and attacked samples. The dataset comprises eight different kinds of attacks: "Bruteforce, DDoS, DoS, Heartbleed, Infiltration, Portscan, and Web". This dataset has 2,491,689 samples in total—2,273,097 normal and 218,592 attack samples. The entire dataset is split into two subsets of 1,744,184 and 747,505 samples, respectively, by dividing it into training and testing sets in a 70:30 ratio. Table 3 provides information on the various class types. Samples from this dataset include 78 features.

d) CIC-DDoS2019: The dataset was gathered for testing and training on two different days. Twelve DDoS attacks are included in the training set; these include DDoS-based attacks from "SNMP, NetBIOS, LDAP, TFTP, NTP, SYN, UDP, WebDDoS, MSSQL, UDPLag, DNS, and SSDP. Seven DDoS assaults against PortScan, SYN, MSSQL, UDP-Lag, LDAP, UDP, and NetBIOS" are included in the testing data. The distribution of the various attacks is displayed in Figure 7. Using CICFlowMeter tools, the researchers retrieved over 80 flow features from the CIC-DDoS2019. The Canadian Institute for Cybersecurity website makes the dataset available to the general public inflow and PCAP file formats [46].

8.2. Preprocessing

The focus of this research is on a binary classification issue for anomaly detection, in which every observation is assigned to either the attack or normal class. We performed the following pre-processing actions on the datasets we had chosen before training the DDoS attack model:

a) Data cleaning: Destination and source IP, flow ID, and Port are the three types of socket information that are different in the CIC-IDS2017 and CIC-DDoS2019 datasets. Since socket-involved features can differ from network to network. Hence, eliminated them from the data samples to solve the overfitting issue. Additionally, we eliminated from the dataset any samples that had the 'NaN' and 'INF' feature values.

b) Data encoding: Aside from the traffic labels of the CIC-DDoS2019 and CIC-IDS2017, respectively, the final dataset includes 77 & 78 other attributes. The categorical features, like protocol type, services, and flag, were converted into numerical features using one-hot encoding. For instance, the mappings for the TCP, UDP, and ICMP protocols are (1,0,0), (0,1,0), and (0,0,1), respectively. Similar to this, numerical features have been mapped to the "flag" feature with 11 values and the "services" feature with 70 values. As a result, 121 numerical features are ultimately created from 41 original features. Binary encoding is also used to translate the non-numerical class labels into numerical categories. These cases are allocated to 1 and 0, respectively, because the only binary classification that we have taken into consideration in our model is to identify the anomalous and regular traffic from input data. A dataset's duplication could cause the anomaly detection model to be biased toward more frequent records during training. To fix this, we eliminated all of the duplicate records from the data and only retained one copy of each entry. Following the process, the CIC-IDS2017 (DoS) dataset's sample count is lowered to 587, 966.

c) Data normalization: The original value scales have been eliminated by normalizing the numerical features. Each feature has undergone Min-Max Normalization, which rescales the feature range to fall inside [0, 1]. The Min-Max Normalization is as shown in the equations (19) as below:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (19)$$

where z_i is the i^{th} normalized data and x_i is a feature.

d) K-fold cross validation: The next experiment will use the k-fold validation methodology to randomly select the dataset and evaluate the effectiveness of the suggested method. The dataset is divided into k subgroups of the same size. In this investigation, a 10-fold subset is employed. There are ten data subsets for each fold when K = 10. The data is divided into 10 folds with roughly equal magnitudes for each fold. On each of the ten data subsets, the cross-validation test is run using a 9-fold training set and a 1-fold testing set.

RESEARCH ARTICLE

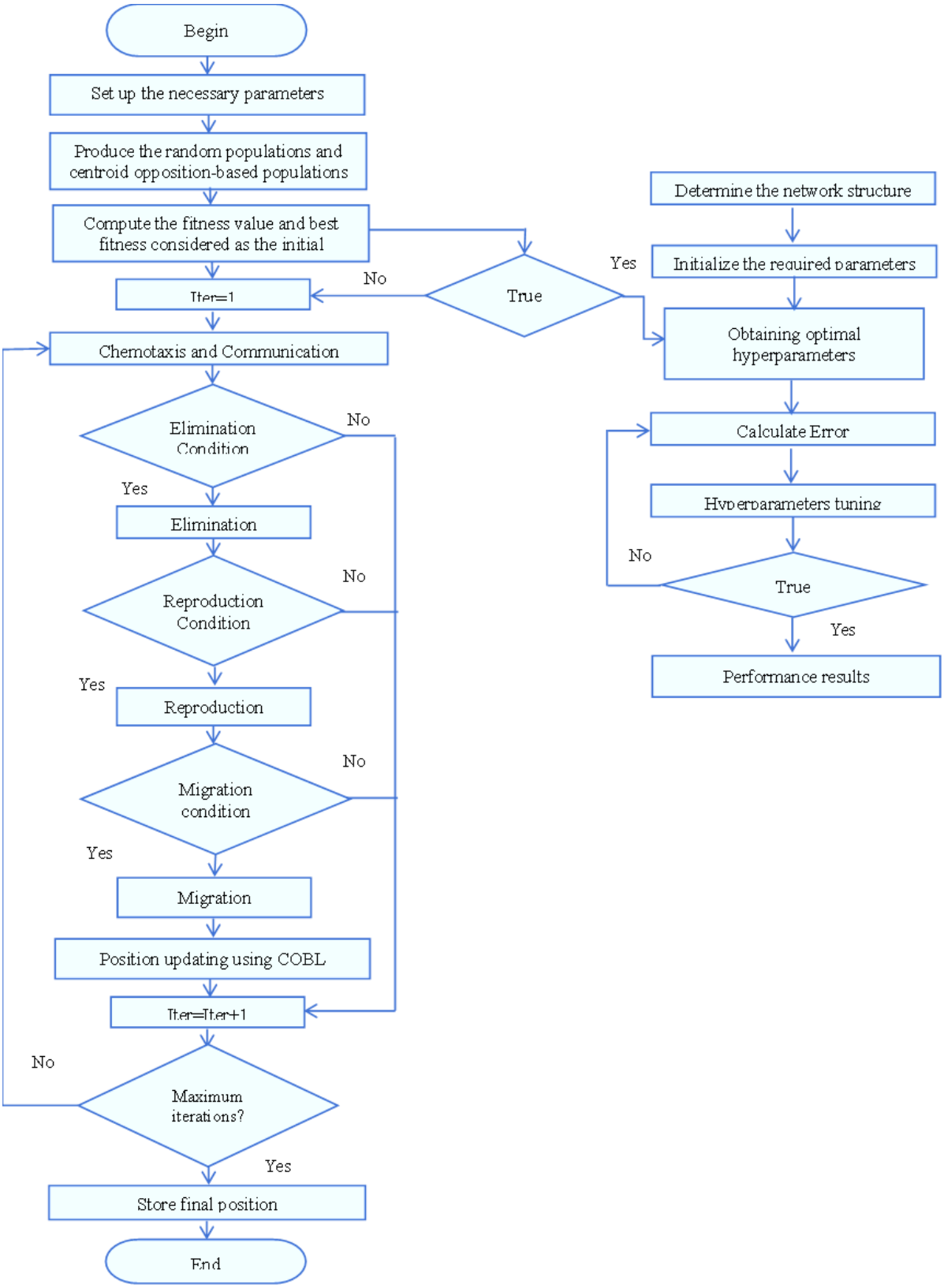

Figure 1 Flowchart for the Proposed Research Work

RESEARCH ARTICLE

8.3. Parameter Optimization

Hyperparameters are crucial in influencing the effectiveness, efficiency, and behavior of the algorithms in both swarm intelligence and deep learning. Elman RNNs can benefit from the efficient application of COBCO for hyperparameter tuning. Elman RNNs are renowned for their capacity to detect temporal dependencies in sequential data; nevertheless, to attain peak performance, hyperparameters must be carefully adjusted. COBCO is an SI optimization algorithm that can help with this process because of its capacity for escape from local optima and adaptable search features. A set of hyperparameters is represented by each bacterium. A bacterium's location inside the search space correlates to particular values of the hyperparameters. With a given set of hyperparameters, the objective function assesses the Elman RNN's performance. An inappropriate selection of ERNN hyperparameters may adversely affect network performance by increasing computation time, causing under- or overfitting, and lowering the convergence rate. Therefore, choosing the ENN's hyperparameters is essential which directly affects the performances of the given solutions.

Three hyperparameters are optimized in this present research work such as learning, weights and biases, and number of neurons. An Elman RNN's ***weights and biases*** are crucial elements that control how the network interprets input, learns from it, captures temporal dependencies, and eventually generates predictions. The selection of weights is between -0.5 and 0. 5. The ***learning rate***, which regulates how much the weights and biases are changed during each training cycle of the neural network, is an essential hyperparameter. The learning rate, which falls between 0.1 and 0.9, is taken into account. An ENN's capacity, learning capability, and overall performance are all impacted by the significance hyperparameter of the number of hidden neurons. An Elman RNN's ability to manage temporal dependencies, learn and represent patterns in the data, and use a minimal amount of processing resources is all greatly impacted by the number of hidden neurons in the network. Achieving a balance between underfitting and overfitting, computational efficiency, and the capacity to generalize well to new data all depend on selecting the appropriate number of hidden neurons. There are between 10 and 100 hidden neurons chosen.

In BCO, parameters are essential for defining how the optimization algorithm behaves and how effective it is. BCO is a technique for solving optimization issues that draws inspiration from the behavior of bacterial colonies found in nature. The parameter of the BCO is shown in Table 2. The convergence rate of BCO is decided based on its chemotaxis step (N_C) values and swim step (N_s). It takes longer to compute the chemotaxis step when it is higher. As a result, the current study chooses a limited number of chemotaxis steps such as $N_C = 100$. Swim step is selected as $N_s = 4$. The reproduction value is designated as ($N_{re} = 4$), and the dispersal step value is designated as $N_{ed} = 2$. Dispersal values, step size, and likelihood of elimination are all crucial elements in figuring out how well the BCO algorithm works. The best performance is defined as having the lowest goal value. The lowest step size value (C_{min}) and greatest step size value (C_{max}) are two distinct step size values. An escape from the local optima problem is an ideal value, and the elimination and dispersal probability P_{ed} value is another important BCO parameter. Therefore, 0.25 is the chosen probability value. The parameter values for the remaining algorithm are set to those found in its reference papers, like k-means [47], ACO [48], PSO [49], BFO [50], and BCO [28].

Table 2 Parameter's Values of BCO and ERNN

ERNN		BCO	
Parameter	Value	Parameter	Value
Activation function	sigmoid TanH	S	100
Objective function	MSE	N_C	100
Learning rate	0.05	N_s	4
Training epochs	1000	N_{re}	4
Error	0.0005	N_{ed}	2
Weight range	-0.5 and 0.5	P_{ed}	0.25
Hidden neurons	10-100	C_{min} and C_{max}	0.1 and 0.4

8.4. Performance Measures

An essential component in detecting DDoS attacks is the study and discussion of the results. A crucial component of DDoS attack detection is results analysis and discussion, which offers insights into the types of attacks, their effects, the limitations of detection, and potential areas for development. The present section discusses the results analysis for analyzing the performance of the proposed methods. To evaluate the created detection method's suitability for comparisons, four distinct performance criteria are taken into account across the four datasets. Performance measurements are important for evaluating how well DDoS attack detection systems work. DDoS detection systems can enhance network security and lessen the impact of DDoS attacks by improving their capacity to precisely and quickly identify and mitigate DDoS attacks. This can be achieved by tracking and adjusting certain performance parameters. Our

RESEARCH ARTICLE

suggested model has been assessed using the following four performance metrics as follows,

8.4.1. Accuracy

DDoS attack detection systems must be accurate in differentiating between malicious activity and legitimate traffic to identify DDoS attacks. The percentage of exactly identified examples among all examples is known as accuracy which is shown in Equation (20),

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \quad (20)$$

Table 3 Training Result Analysis for Accuracy

Methods	NSL-KDD	UNSW-NB15	CICIDS2017	CIC-DDoS2019
COBCO+ENN	97.11	98.22	98.11	98.95
BCO+ENN	95.14	97.19	97.49	97.13
IPSO +ENN	94.43	96.47	96.77	95.06
PSO+ENN	92.75	94.76	95.74	94.37
GA+ENN	90.15	93.11	93.73	91.78
ENN	85.77	92.49	91.79	88.84
BPNN	82.48	87.47	90.17	85.45

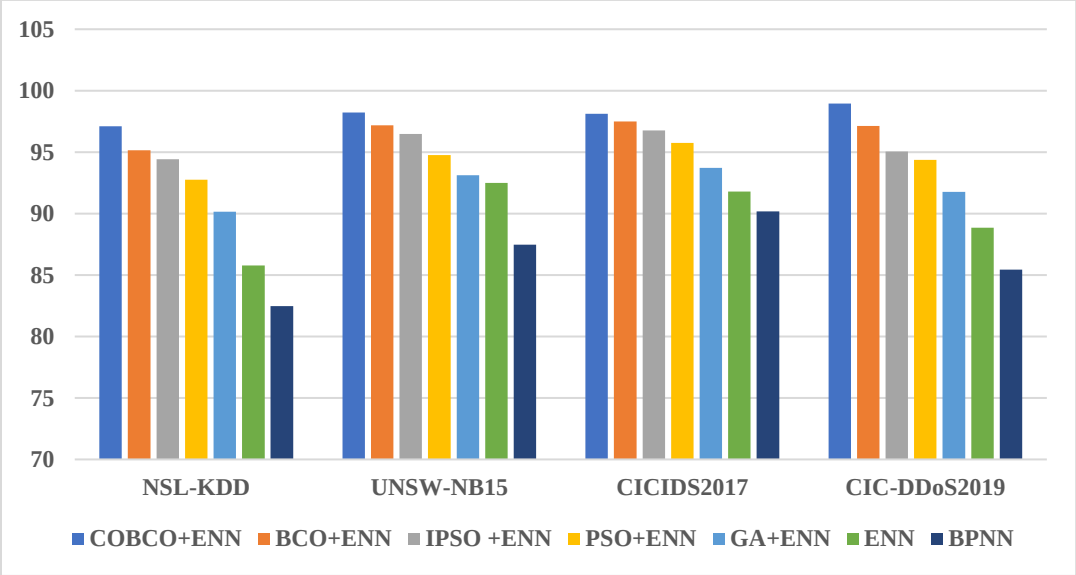

Figure 2 Training Result Analysis for Accuracy

The accuracy-based performance training results for all datasets, including NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019, are displayed in Table 3 and Figure 3, respectively. Table 3 and Figure 3 show that, when compared to alternative approaches, the suggested method produced high accuracy for all datasets, including 97.11 percent accuracy for NSL-KDD, 98.22 percent accuracy for UNSW-NB15, 98.11 percent accuracy for CICIDS2017, and 98.95 percent accuracy for CIC-DDoS2019. Figure 4 shows that, when compared to alternative approaches, the suggested method produced high testing accuracy for all datasets.

RESEARCH ARTICLE

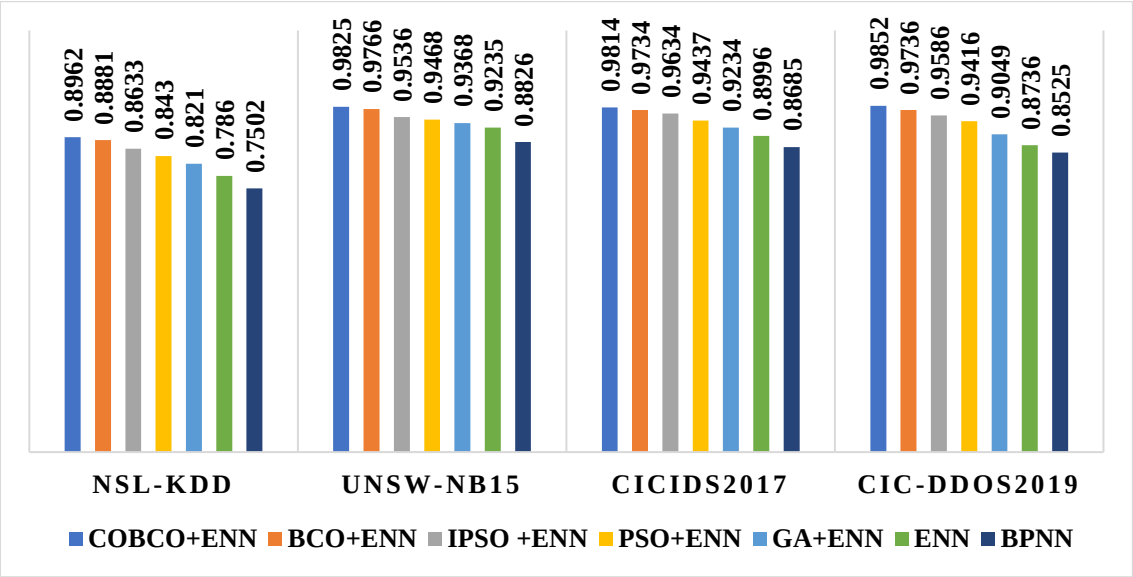

Figure 3 Testing Result Analysis for Accuracy

8.4.2. Precision

Precision, which gauges the system's capacity to accurately detect harmful activity without incorrectly classifying benign traffic, is a crucial performance metric in DDoS attack detection. Precision becomes more crucial in situations when reducing false alarms is crucial. The percentage of real positives among all predicted positives is measured by precision which is shown in Equation (21).

$$Precision = \frac{TP}{TP + FP} \times 100 \quad (21)$$

Table 4 Training Result Analysis for Precision

Methods	NSL-KDD	UNSW-NB15	CICIDS2017	CIC-DDoS2019
COBCO+ENN	98.17	99.76	98.43	98.51
BCO+ENN	97.47	97.21	97.19	97.34
IPSO +ENN	96.80	96.44	96.44	96.46
PSO+ENN	95.76	95.75	95.48	94.76
GA+ENN	94.11	93.21	93.89	93.64
ENN	93.47	92.43	92.47	91.27
BPNN	91.71	90.16	90.23	88.23

The precision-based performance results for all datasets, including NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019, are displayed in Table 4 and Figure 5, respectively. Table 4 and Figure 5 show that, when compared to alternative approaches, the suggested method produced high accuracy for all datasets, including 98.17 percent accuracy for NSL-KDD, 99.76 percent accuracy for UNSW-NB15, 98.43 percent accuracy for CICIDS2017, and 98.51 percent accuracy for CIC-DDoS2019. Figure 6 shows that, when compared to alternative approaches, the suggested method produced high testing accuracy for all datasets.

RESEARCH ARTICLE

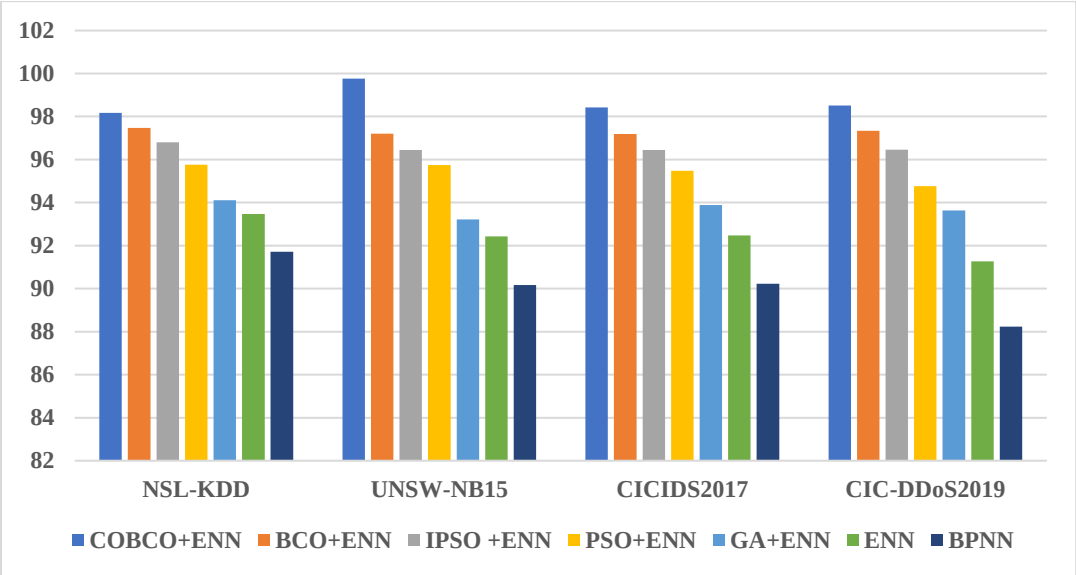

Figure 4 Training Result Analysis for Precision

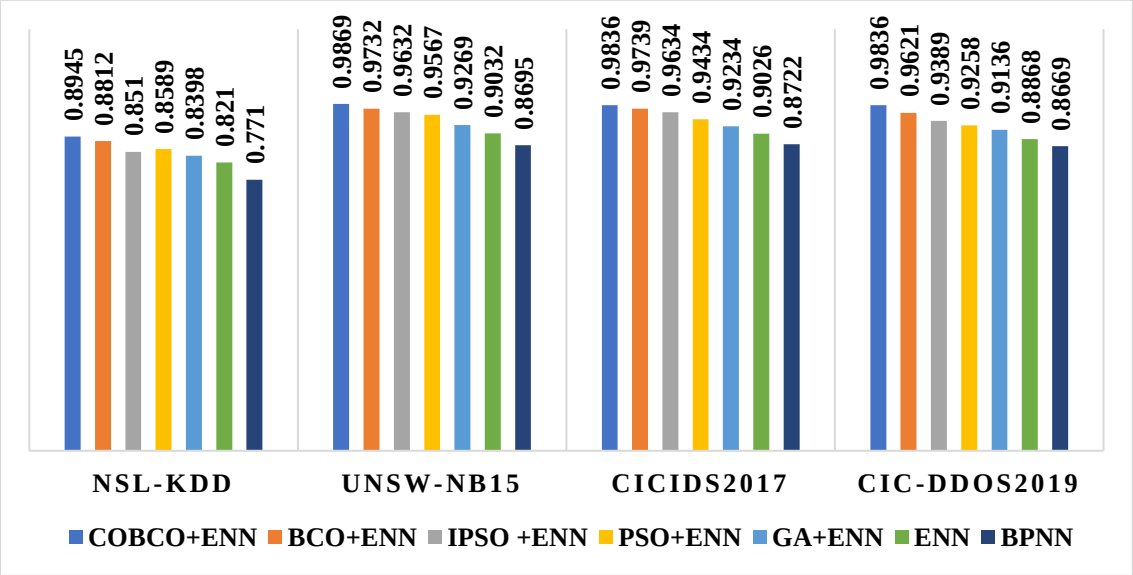

Figure 5 Testing Result Analysis for Precision

8.4.3. Recall

Recall is a vital parameter, especially in situations where it is imperative to detect every possible danger when assessing the effectiveness of DDoS attack detection systems. Recall quantifies the percentage of accurately detected true positives which is shown in Equation (22)

$$Recall = \frac{TP}{TP+FN} \times 100 \quad (22)$$

The recall-based performance results for all datasets, including NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019, are displayed in Table 5 and Figure 7, respectively.

Table 5 and Figure 7 shows that, when compared to alternative approaches, the suggested method produced high accuracy for all datasets, including 98.14 percent accuracy for NSL-KDD, 98.10 percent accuracy for UNSW-NB15, 97.96 percent accuracy for CICIDS2017, and 99.08 percent accuracy for CIC-DDoS2019.

Figure 8 shows that, when compared to alternative approaches, the suggested method produced high testing accuracy for all datasets.

RESEARCH ARTICLE

Table 5 Training Result Analysis for Recall

Methods	NSL-KDD	UNSW-NB15	CICIDS2017	CIC-DDoS2019
COBCO+ENN	98.14	98.10	97.96	99.08
BCO+ENN	97.76	97.49	96.41	97.28
IPSO +ENN	96.88	96.47	95.74	95.37
PSO+ENN	96.65	95.63	94.63	94.09
GA+ENN	94.00	93.22	93.14	92.85
ENN	91.16	92.17	92.77	90.57
BPNN	88.79	90.71	91.47	87.28

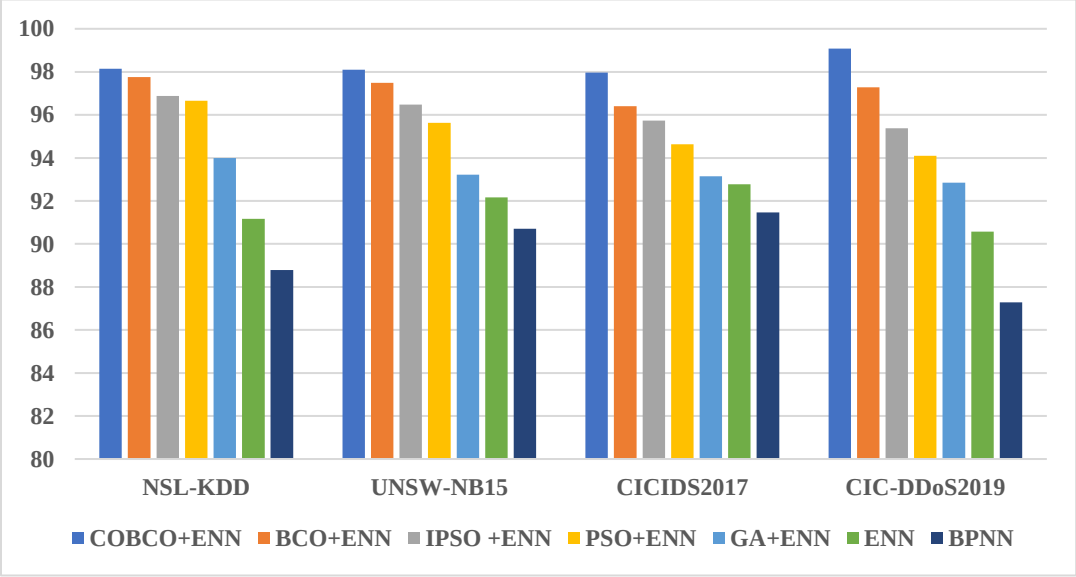

Figure 6 Training Result Analysis for Recall

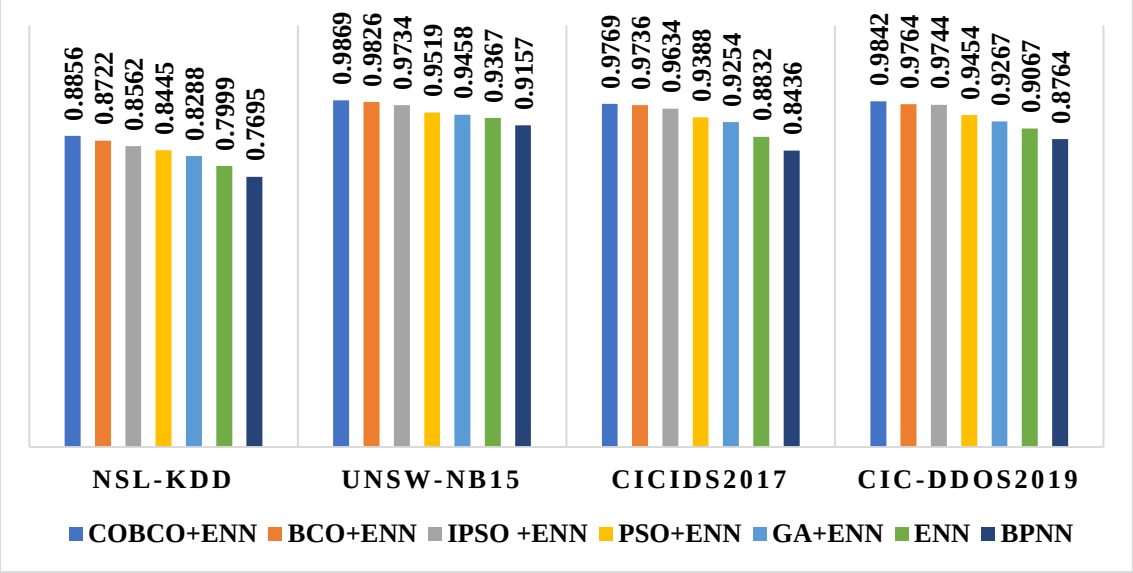

Figure 7 Testing Result Analysis for Recall

RESEARCH ARTICLE

8.4.4. F1-Score

An essential indicator for assessing the effectiveness of DDoS attack detection systems is the F-score, particularly in situations where recall and precision must be balanced. It offers a more thorough evaluation of the detection system's efficacy by combining these two indicators into a single score. It is shown as Equation (23)

$$F - Score = 2 \times \frac{Pr\,ecision \times Recall}{Pr\,ecision + Recall} \times 100 \quad (23)$$

Table 6 Training Result Analysis for F-Score

Methods	NSL-KDD	UNSW-NB15	CICIDS2017	CIC-DDoS2019
COBCO+ENN	99.10	98.16	98.29	98.07
BCO+ENN	98.63	97.40	97.52	96.82
IPSO +ENN	97.10	96.46	95.74	95.17
PSO+ENN	96.77	95.33	94.16	93.56
GA+ENN	95.47	94.19	93.78	92.97
ENN	93.05	92.74	92.79	90.28
BPNN	92.77	89.58	90.14	88.14

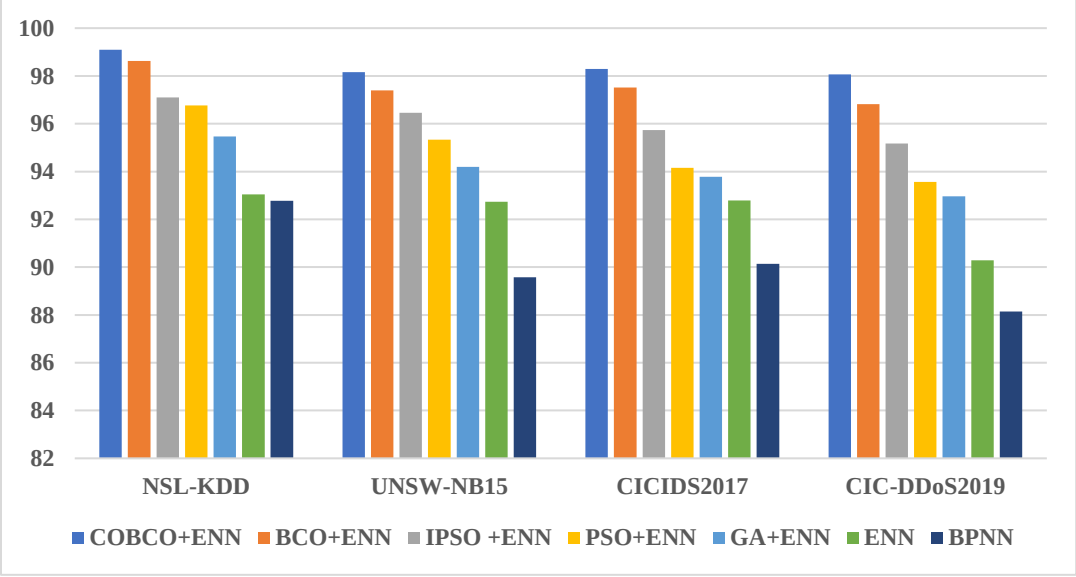

Figure 8 Training Result Analysis for F-Score

The F-Score-based performance results for all datasets, including NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019, are displayed in Table 6 and Figure 9 respectively.

Table 6 and Figure 9 show that, when compared to alternative approaches, the suggested method produced high accuracy for all datasets, including 98.10 percent accuracy for NSL-KDD, 98.16 percent accuracy for UNSW-NB15, 98.29 percent accuracy for CICIDS2017, and 99.07 percent accuracy for CIC-DDoS2019.

Figure 10 shows that, when compared to alternative approaches, the suggested method produced high testing accuracy for all datasets. From the equations (20), (21), (22), and (23), TP (True positive): An attack instance is measured as TP if it is accurately characterized.

FP (False positive): An attack is measured when a normal instance is classified as such. TN (True negative): A normal occurrence is measured if it is classified as normal. FN (False negative): An attack instance is measured as FN if it is classified as normal.

RESEARCH ARTICLE

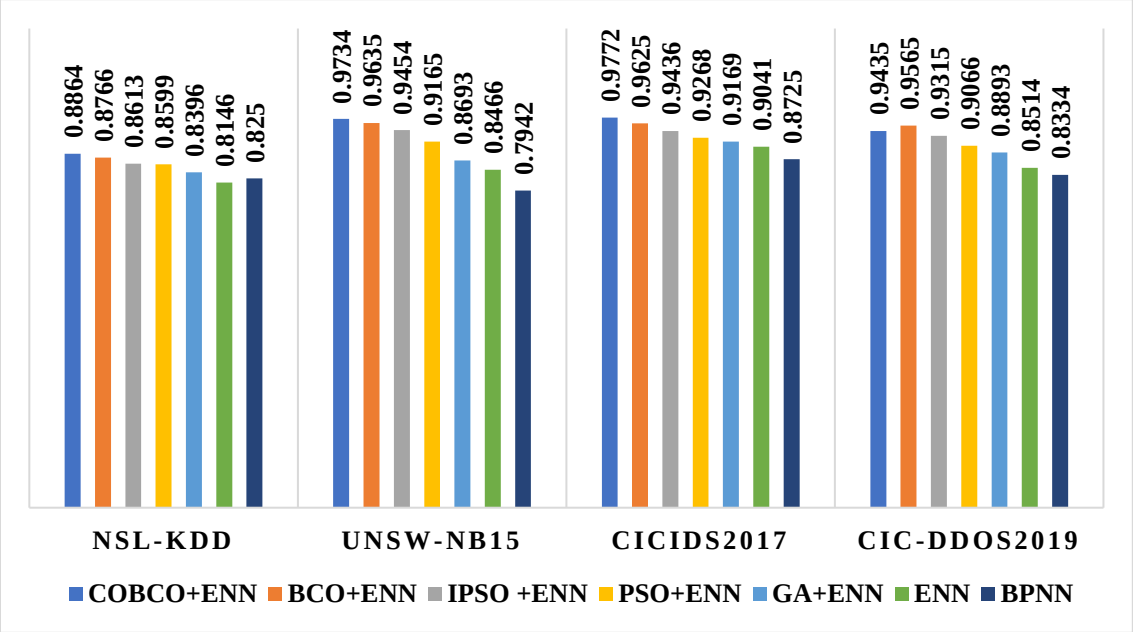

Figure 9 Testing Result Analysis for F-Score

8.4.5.

Convergence Analysis

To analyze the performance of optimized ENN methods, convergence analysis is also considered in the present work. Convergence analysis is an essential component that is vital to comprehending and assessing detection model performance for multiple reasons. Understanding the behavior and effectiveness of ENN in DDoS attack detection tasks is crucial for making sure the model learns efficiently, generalizes well, and produces accurate results. This is where convergence analysis comes in. The convergence analysis is conducted on the loss as it varies over epochs. Plot the number of epochs versus the training loss (MSE). Finding out whether the optimized ENN achieves a consistent performance level and how well it can learn to detect DDoS attacks are revealed by performing a convergence analysis. The convergence study of DDoS attack detection techniques for all datasets, including NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019, can be observed in Figures 11, 12, 13, and 14.

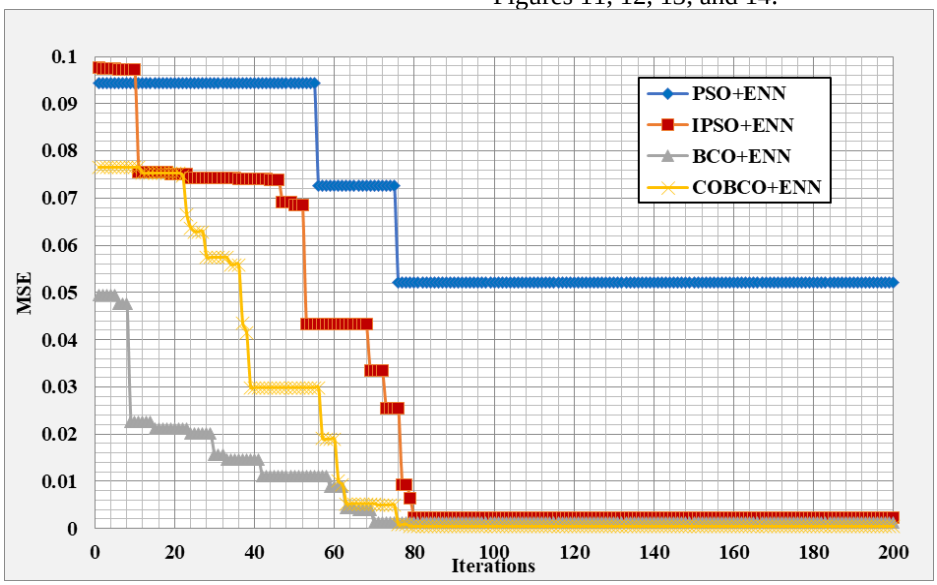

Figure 10 Convergence Analysis of Detection Method for NSL-KDD

RESEARCH ARTICLE

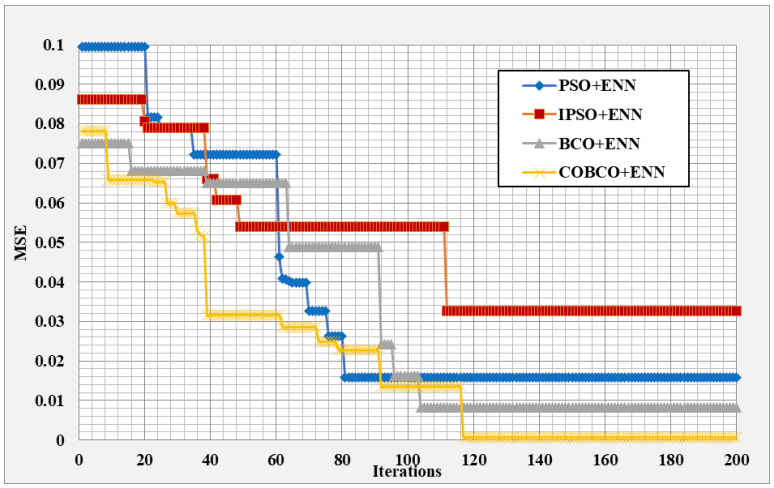

Figure 11 Convergence Analysis of Detection Method for UNSW-NB15

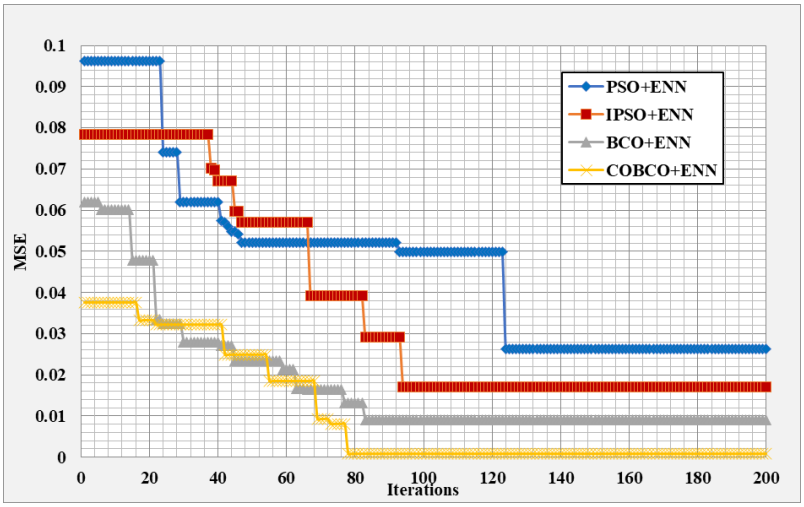

Figure 12 Convergence Analysis of Detection Method for CICIDS2017

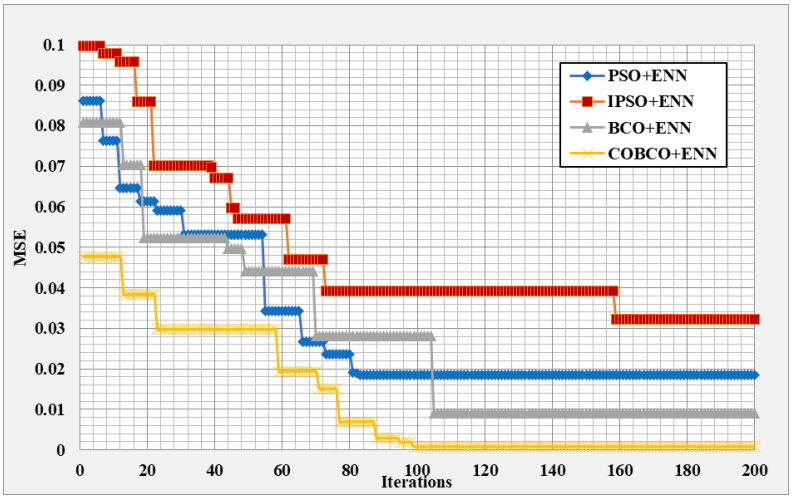

Figure 13 Convergence Analysis of Detection Method for CIC-DDoS2019

RESEARCH ARTICLE

According to the experiment results, the combination of COBCO with Elman RNNs presents a viable method for DDoS attack detection. The process of optimization improves the network's capacity to quickly and precisely identify attacks, offering strong security for cloud services. To fully achieve its potential, this approach's actual application necessitates careful consideration of computational resources and data quality. When COBCO is used to optimize the ENN, the accuracy of DDoS attack detection can be greatly increased. More accurate detection models are produced by adjusting the neural network's hyperparameters with the aid of COBCO. By identifying intricate patterns in network traffic that point to possible DDoS attacks, this optimization can increase the total detection rate beyond that of conventional techniques. Overall performance evaluation shows that the more adaptive real-time detector of the new and improved ERNN technology can analyse dangerous data from incoming data.

9. CONCLUSIONS

In this work, we optimized the ENN using CBCO which presents a unique method for detecting attacks. The model's capacity to traverse the intricate solution space was greatly improved by the integration of CBCO with ENN, which resulted in quicker convergence and better accuracy in identifying different DDoS attack patterns. By producing a wide range of potential solutions, COBL enhanced the detection framework and reduced the drawbacks of conventional OBL methods. Our comprehensive simulation results presented that the recommended method performs better than the state-of-the-art DDoS detection methods, especially when it comes to accuracy and detection rate. The improved performance is ascribed to the complementary abilities of BCO's optimization skills and COBL's capacity to infuse the ENN with more resilient learning dynamics. Overall, the results point to a promising direction for the development of DDoS detection techniques in cloud computing, providing a very practical and scalable answer to security issues facing the industry. This is because BCO and COBL work well together. Subsequent research endeavors will center around enhancing the model, investigating its suitability for various categories of cyber hazards, and verifying the methodology in authentic cloud systems.

REFERENCES

[1] M. Alam, M. Shahid, and S. Mustajab, "Security challenges for workflow allocation model in cloud computing environment: a comprehensive survey, framework, taxonomy, open issues, and future directions," The Journal of Supercomputing, pp. 1-65, 2024.

[2] D. N, J. Katiravan, and S. S.P, "Botnet Attack Detection in IoT Devices using Ensemble Classifiers with Reduced Feature Space," International Research Journal of Multidisciplinary Technovation, vol. 6, no. 3, pp. 274-295, 05/22 2024, doi: 10.54392/irjmt24321.

[3] S. K. V, M. K. V, C. N. Azmea, and K. K. Vaigandla, "BCSDNCC: A Secure Blockchain SDN framework for IoT and Cloud Computing," International Research Journal of Multidisciplinary Technovation, vol. 6, no. 3, pp. 26-44, 04/16 2024, doi: 10.54392/irjmt2433.

[4] P. Ravi Kiran Varma, S. RR, and M. Vanitha, "Enhanced Elman spike neural network based intrusion attack detection in software defined Internet of Things network," Concurrency and Computation: Practice and Experience, vol. 35, no. 2, p. e7503, 2023.

[5] M. T. Hussan, G. V. Reddy, P. Anitha, A. Kanagaraj, and P. Naresh, "DDoS attack detection in IoT environment using optimized Elman recurrent neural networks based on chaotic bacterial colony optimization," Cluster Computing, pp. 1-22, 2023.

[6] B. Sivasakthi and D. Selvanayagi, "Prediction of osteoporosis disease using enhanced Elman recurrent neural network with bacterial colony optimization," in Computational Vision and Bio-Inspired Computing: Proceedings of ICCVBIC 2022: Springer, 2023, pp. 211-220.

[7] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 3, pp. 825-831, 2022.

[8] A. V. Kachavimath and D. Narayan, "A deep learning-based framework for distributed denial-of-service attacks detection in cloud environment," in Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 1, 2021: Springer, pp. 605-618.

[9] Y. Sanjalawe and T. Althobaiti, "DDoS Attack Detection in Cloud Computing Based on Ensemble Feature Selection and Deep Learning," Computers, Materials & Continua, vol. 75, no. 2, 2023.

[10] T. Khempetch and P. Wuttidittachotti, "DDoS attack detection using deep learning," IAES International Journal of Artificial Intelligence, vol. 10, no. 2, p. 382, 2021.

[11] D. Kumar, R. Pateriya, R. K. Gupta, V. Dehalwar, and A. Sharma, "DDoS detection using deep learning," Procedia Computer Science, vol. 218, pp. 2420-2429, 2023.

[12] S. Potluri, M. Mangla, S. Satpathy, and S. N. Mohanty, "Detection and prevention mechanisms for DDoS attack in cloud computing environment," in 2020 11th international conference on computing, communication and networking technologies (ICCCNT), 2020: IEEE, pp. 1-6.

[13] A. Agarwal, M. Khari, and R. Singh, "Detection of DDOS attack using deep learning model in cloud storage application," Wireless Personal Communications, pp. 1-21, 2022.

[14] 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, p. 114520, 2021.

[15] 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 & Theoretical Artificial Intelligence, vol. 33, no. 3, pp. 405-424, 2021.

[16] A. Amjad, T. Alyas, U. Farooq, and M. A. Tariq, "Detection and mitigation of DDoS attack in cloud computing using machine learning algorithm," EAI Endorsed Transactions on Scalable Information Systems, vol. 6, no. 23, pp. e7-e7, 2019.

[17] S. Ur Rehman et al., "DIDDOS: An approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU)," Future Generation Computer Systems, vol. 118, pp. 453-466, 2021.

[18] D. Alghazzawi, O. Bamasag, H. Ullah, and M. Z. Asghar, "Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection," Applied Sciences, vol. 11, no. 24, p. 11634, 2021.

[19] A. V. Songa and G. R. Karri, "Ensemble-RNN: A Robust Framework for DDoS Detection in Cloud Environment," Majlesi Journal of Electrical Engineering, vol. 17, no. 4, pp. 31-44, 2023.

[20] S. Balasubramaniam et al., "Optimization enabled deep learning-based DDoS attack detection in cloud computing," International Journal of Intelligent Systems, vol. 2023, 2023.

RESEARCH ARTICLE

[21] G. S. Kushwah and V. Ranga, "Optimized extreme learning machine for detecting DDoS attacks in cloud computing," Computers & Security, vol. 105, p. 102260, 2021.
[22] P. T. Dinh and M. Park, "R-EDoS: robust economic denial of sustainability detection in an SDN-based cloud through stochastic recurrent neural network," IEEE Access, vol. 9, pp. 35057-35074, 2021.
[23] S. Sumathi, R. Rajesh, and S. Lim, "Recurrent and deep learning neural network models for DDoS attack detection," Journal of Sensors, vol. 2022, 2022.
[24] J. L. Elman, "Finding structure in time," Cognitive science, vol. 14, no. 2, pp. 179-211, 1990.
[25] B. Niu and H. Wang, "Bacterial colony optimization: principles and foundations," in Emerging Intelligent Computing Technology and Applications: 8th International Conference, ICIC 2012, Huangshan, China, July 25-29, 2012. Proceedings 8, 2012: Springer, pp. 501-506.
[26] K. Vijayakumari and V. Baby Deepa, "Fuzzy C-means hybrid with fuzzy bacterial colony optimization," in Advances in electrical and computer technologies: select proceedings of ICAECT 2020, 2021: Springer, pp. 75-87.
[27] V. Prakash, V. Vinothina, K. Kalaiselvi, and K. Velusamy, "An improved bacterial colony optimization using opposition-based learning for data clustering," Cluster Computing, vol. 25, no. 6, pp. 4009-4025, 2022.
[28] J. Revathi, V. Eswaramurthy, and P. Padmavathi, "Bacterial colony optimization for data clustering," in 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2019: IEEE, pp. 1-4.
[29] J. Revathi, V. Eswaramurthy, and P. Padmavathi, "Hybrid data clustering approaches using bacterial colony optimization and k-means," in IOP Conference Series: Materials Science and Engineering, 2021, vol. 1070, no. 1: IOP Publishing, p. 012064.
[30] K. Tamilarisi, M. Gogulkumar, and K. Velusamy, "Data clustering using bacterial colony optimization with particle swarm optimization," in 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2021: IEEE, pp. 1-5.
[31] S. S. Babu and K. Jayasudha, "A simplex method-based bacterial colony optimization for data clustering," in Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2021: Springer, 2022, pp. 987-995.
[32] S. S. Babu and K. Jayasudha, "A simplex method-based bacterial colony optimization algorithm for data clustering analysis," International Journal of Pattern Recognition and Artificial Intelligence, vol. 36, no. 12, p. 2259027, 2022.
[33] H. Wang, L. Tan, and B. Niu, "Feature selection for classification of microarray gene expression cancers using Bacterial Colony Optimization with multi-dimensional population," Swarm and Evolutionary Computation, vol. 48, pp. 172-181, 2019.
[34] S. İlkin, T. H. Gençtürk, F. K. Gülağız, H. Özcan, M. A. Altuncu, and S. Şahin, "hybSVM: Bacterial colony optimization algorithm based SVM for malignant melanoma detection," Engineering Science and Technology, an International Journal, vol. 24, no. 5, pp. 1059-1071, 2021.
[35] B. Niu, T. Xie, Y. Bi, and J. Liu, "Bacterial colony optimization for integrated yard truck scheduling and storage allocation problem," in Intelligent Computing in Bioinformatics: 10th International Conference, ICIC 2014, Taiyuan, China, August 3-6, 2014. Proceedings 10, 2014: Springer, pp. 431-437.
[36] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), 2005, vol. 1: IEEE, pp. 695-701.
[37] S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad, and G. F. Naterer, "Computing opposition by involving entire population," in 2014 IEEE congress on evolutionary computation (CEC), 2014: IEEE, pp. 1800-1807.
[38] R. S. Parpinelli, G. F. Plichoski, R. S. D. Silva, and P. H. Narloch, "A review of techniques for online control of parameters in swarm intelligence and evolutionary computation algorithms," International Journal of Bio-Inspired Computation, vol. 13, no. 1, pp. 1-20, 2019.
[39] L. Yang, F. Wang, J. Zhang, and W. Ren, "Remaining useful life prediction of ultrasonic motor based on Elman neural network with improved particle swarm optimization," Measurement, vol. 143, pp. 27-38, 2019.
[40] Y. Wang, L. Wang, F. Yang, W. Di, and Q. Chang, "Advantages of direct input-to-output connections in neural networks: The Elman network for stock index forecasting," Information Sciences, vol. 547, pp. 1066-1079, 2021.
[41] A. Sadeghi-Niaraki, P. Mirshafiei, M. Shakeri, and S.-M. Choi, "Short-term traffic flow prediction using the modified elman recurrent neural network optimized through a genetic algorithm," IEEE Access, vol. 8, pp. 217526-217540, 2020.
[42] N. Chowdhury, "A comparative analysis of feed-forward neural network & recurrent neural network to detect intrusion," in 2008 International Conference on Electrical and Computer Engineering, 2008: IEEE, pp. 488-492.
[43] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection," Computers & Security, vol. 75, pp. 36-58, 2018.
[44] T. A. Tuan, H. V. Long, L. H. Son, R. Kumar, I. Priyadarshini, and N. T. K. Son, "Performance evaluation of Botnet DDoS attack detection using machine learning," Evolutionary Intelligence, vol. 13, no. 2, pp. 283-294, 2020.
[45] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 Military Communications and Information Systems Conference (MilCIS), 10-12 Nov. 2015 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.
[46] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in 2019 international carnahan conference on security technology (ICCST), 2019: IEEE, pp. 1-8.
[47] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," Pattern recognition, vol. 36, no. 2, pp. 451-461, 2003.
[48] P. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," Analytica Chimica Acta, vol. 509, no. 2, pp. 187-195, 2004.
[49] I. De Falco, A. Della Cioppa, and E. Tarantino, "Facing classification problems with particle swarm optimization," Applied Soft Computing, vol. 7, no. 3, pp. 652-658, 2007.
[50] M. Wan, L. Li, J. Xiao, C. Wang, and Y. Yang, "Data clustering using bacterial foraging optimization," Journal of Intelligent Information Systems, vol. 38, no. 2, pp. 321-341, 2012.

Authors

Mrs. S. Kalvikkarasi is working as an Assistant professor, PG and Research Department of Computer Science, Government Arts College, Trichy-22. She has 19 years of teaching experience and has also presented papers at various international conferences. She is pursuing a Ph.D(Computer Science) Part-time at Government Arts College, Karur.

Dr. A. Saraswathi is working as an Associate professor, PG and Research Department of Computer Science, Government Arts College, Karur. She has 23 years of Teaching experience and 10 years of experience in Research. She has published more than 15 international publications in various journals.

RESEARCH ARTICLE

How to cite this article:

S. Kalvikkarasi, A. Saraswathi, "DDoS Attack Detection in Cloud Computing Using Optimized Elman Neural Network Based on Bacterial Colony Optimization and Centroid Opposition-Based Learning", International Journal of Computer Networks and Applications (IJCNA), 11(6), PP: 835-854, 2024, DOI: 10.22247/ijcna/2024/50.