Journal of Computer Science

Original Research Paper

The Optimized Extreme Learning Machine (GA-OELM) for DDoS Attack Detection in Cloud Environment

Meryem Ec-Sabery, Adil Ben Abbou, Abdelali Boushaba, Fatiha Mrabti and Rachid Ben Abbou

Department of Computer Science, Faculty of Sciences and Technology, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Article history
Received: 03-08-2024
Revised: 08-10-2024
Accepted: 14-10-2024

Corresponding Author:
Meryem Ec-Sabery
Department of Computer Science, Faculty of Sciences and Technology, Sidi Mohamed Ben Abdellah University, Fez, Morocco
Email: meryem.ecsabery@usmba.ac.ma

Abstract: The widespread adoption of cloud computing has increased the attack surface and raised significant security concerns. A Distributed Denial of Service (DDoS) is a serious attack that depletes the network and server resources in cloud computing, causing service downtime or reduced performance. Therefore, defending against DDoS attacks becomes an urgent need. In this present paper, we propose an Optimized Extreme Learning Machine based on Genetic Algorithm (GA-OELM) for detecting DDoS attack patterns. The proposed model uses an improved GA for optimizing the weights and biases of the ELM hidden layer. The experiment is evaluated using three datasets namely, CICDDOS2019, NSL-KDD, and UNSW-NB15, and proves that the detection performance of the proposed GA-OELM is better than the classic ELM model and some state of art techniques.

Keywords: DDoS Attack, Extreme Learning Machine, Genetic Algorithm, Cloud Computing

Introduction

Cloud computing (Cloud, 2011) is an internet-based platform that provides and shares resources like storage, computation, and networking among multiple tenants. These tenants can flexibly meet their IT needs while paying only for the resources they consume. As the demand for cloud services grows, cloud administrators face the challenging task of ensuring both the security and availability of cloud services (Syed *et al*., 2017). However, due to its distributed and open-access architecture, cloud platforms are susceptible to various cyber threats. According to the Cloud Security Alliance (CSA) report (Jon-Michael and Greg, 2020), DDoS attack is one of the top eleven attacks that frequently hunt the cloud environment. In a DDoS attack, the attackers remotely command a large network of infected machines, known as a botnet, and send an overwhelming volume of malicious traffic to the targeted system (Agrawal and Tapaswi, 2019; Selamat *et al*., 2019). This coordinated attack aims to exhaust the available resources of the cloud such as bandwidth and processing capacity, thereby disrupting its normal functioning (Ouhssini *et al*., 2024). Due to the elasticity feature of the cloud, which automatically adds computational resources, a DDoS attack does not always lead to a downtime in service but it can significantly strain resources, escalate the costs, and impact the overall system efficiency (Kumar *et al*., 2024). Moreover, in public cloud environments, where resources are shared among multiple tenants, there is a risk of collateral damage to non-targets by harming their services and causing autoscaling of their resources as well (Somani *et al*., 2017). This makes mitigating DDoS attacks a critical priority for ensuring the reliability and security of cloud-based systems (Balarezo *et al*., 2022).

DDoS detection systems are typically categorized into two principal approaches (Lata and Singh, 2022): Signature-based detection and anomaly-based detection. Signature-based detection (Hubballi and Suryanarayanan, 2014) works by identifying signatures associated with known DDoS attacks, making it effective for recognizing known threats quickly. However, it may struggle with unknown and zero-day attacks (Canfora *et al*., 2015). In contrast, anomaly-based detection monitors traffic for deviations from the baseline of normal behavior (Zhao *et al*., 2024). Researchers are closely interested in studying anomaly-based detection because it can identify previously unknown or new attacks, particularly in cloud environments where a large number of new DDoS attacks may emerge daily. This approach, while more flexible, can generate a high number of false positives. Requiring tuning to minimize the false alerts and optimize the time of detection.

Anomaly-based detection of DDoS attacks has been proposed by Alqarni (2022); Kumar *et al*. (2023); Velliangiri *et al*. (2021); Songa and Karri (2024);

Wardana *et al*. (2024); Lin *et al*. (2023); Shanmugapriya *et al*. (2024) and relies on the use of machine learning models or deep learning models. Mostly for the cloud, researchers employ neural networks to analyze large volumes of network traffic. Techniques such as Extreme Learning Machines (ELM), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs) can significantly enhance DDoS detection by analyzing traffic patterns in different ways. CNNs (Zoppi *et al*., 2024; Vibhute and Nakum, 2024; Alsoufi *et al*., 2024) excel at detecting DDoS traffic by analyzing network data as images or matrices and identifying key features of attacks. RNNs, especially Long Short-Term Memory (LSTM) (Efendi *et al*., 2024; Kumar *et al*., 2023), are effective for sequential data analysis to recognize unusual spikes or trends that signal DDoS attacks. The big challenge with these techniques lies in the complexity of processing vast amounts of network traffic, which demands substantial computational resources and careful tuning of hyperparameters to achieve optimal performance, making the process time-consuming and requiring high expertise. While the ELM technique is suitable for detecting DDoS attacks due to its rapid training and low computational requirements, it enables quick adaptation to change traffic patterns, particularly in dynamic environments like cloud computing.

ELM (Abu Al-Haija *et al*., 2024; Wang *et al*., 2022a) belongs to a class of Artificial Neural Networks (ANNs), which has only one hidden layer. It is trained in one step, where the biases and the weights of connections between input and hidden layers are initialized at random and the weights connecting hidden and output layers are computed using Moore–Penrose inverse. However, a challenge associated with using the ELM technique is the selection of appropriate hyperparameters, specifically the weights and biases. A random initialization often does not yield optimal performance. To address this issue, many soft computing techniques, such as GA (Mitchell, 2016), Ant Colony Optimization (ACO) (Dorigo and Stutzle, 2019), Particle Swarm Optimization (PSO) (Wang *et al*., 2018), and Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007), for selecting ELM hyperparameters. Nonetheless, these optimization techniques have inherent drawbacks, including premature convergence and limited exploration capabilities. In our paper, we used an improved version of GA, The GA algorithm is inspired by natural evolution, it imitates the process of biological selection to solve optimization problems. The improved GA enhances the exploration of the search space within populations, helping to avoid local optima and increasing the efficiency of the ELM model. Our aim is to identify DDoS attacks in cloud computing using GA-OELM with high accuracy, minimal false positives, and a short detection time.

Related Works

Researchers aim to ensure an effective DDoS detection mechanism in cloud computing based on machine and deep learning techniques. Several articles are closely related to our study in the following literature.

Kumar *et al*. (2023) proposed an LSTM model to detect DDoS attacks; LSTM is a deep learning technique that involves feature extraction and selection algorithms. They used the 17 relevant features from the CICDDoS2019 dataset for training the model. The proposed LSTM reached an accuracy of 98% and outperformed the K-Nearest Neighbor (KNN) and ANN algorithms.

Alqarni, (2022) presented an ensemble technique for preventing DDoS attacks in cloud computing. The proposed ensemble technique includes four classifiers: Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), and KNN and at the end, a majority voting algorithm is applied to combine predictions produced by these classifiers. In a majority vote, the class that receives the highest number of votes is chosen. After the preprocessing phase, 15 features from 88 are selected through the chi-squared method, and part of the CICDDOS2019 dataset is used to train the DDoS detection system. The model achieved 98% accuracy and outperformed each classifier alone.

Kushwah and Ranga, (2021) suggested an improved evolutionary ELM to identify DDoS attacks in cloud computing. SaE-ELM system determined the optimal biases and weights to achieve high accuracy for the ELM algorithm by means of two features: Using a collection of crossover types instead of only one and using different neurons in the hidden layer. The proposed model used four known datasets for training. The performance of the model is better than DT, SVM, and ANN performances.

Kushwah and Ranga (2020) proposed a DDoS detection system for cloud computing based on an ensemble technique (V-ELM), which contains a varied number of ELM machines (4-56 ELMs), and the final prediction is decided by using a majority voting scheme. The number of neurons used in the hidden layer of the ELM machine is up to 1000. The authors used ISCX and NSL-KDD datasets to evaluate the proposed system and compared the performance to adaboost, random forest, ANN, and Back Propagation Neural Network (BPNN).

Lin *et al*. (2023) proposed the integration of a Multi-Feature Extraction (MFE) ELM model into the cloud nodes for detecting and identifying network intrusions. The NSLKDD dataset was utilized for training the model, with comprehensive steps including data preprocessing, feature engineering, and model training. The experimental results indicate that the proposed MFE-ELM algorithm achieved an accuracy of 96.53%.

Velliangiri *et al*. (2021) developed Taylor-Elephant Herd Optimisation based on the Deep Belief Network model to detect DDoS attacks. The proposed model first

gathers log information from each device in the cloud and then creates its log file. Next, it extracts the necessary features from this file, and by using Bhattacharya distance, only key features are selected for classification to decrease the training time. TEHO-DBN is adopted to train the system. Its accuracy is higher, compared to that of NN, Ensemble, and SVM.

Ali and Kushwah (2019) presented a DDoS detection system, which is an ELM with 613 neurons in a hidden layer. The performance of the model is assessed on 15097 samples of the KDD-NSL dataset and compared to the BPNN model with 20 neurons.

Arunadevi and Sathya (2022) introduced an optimized BPNN model to detect DDoS attacks in cloud platforms. The model leverages the Artificial Plant algorithm to optimize the weights and biases of BPNN connections. The proposed detection system is evaluated using four widely recognized datasets: CIC-IDS 2017, NSL-KDD, ISCX-IDS 2012, and UNSW-NB15. The results demonstrate that the APO-BPNN system outperformed traditional BPNN-based detection systems.

Hussein Hadi (2024) introduced a CNN-based model for DDoS detection, organized into three primary stages: Data preprocessing, hyperparameters optimization, and classification. The model requires extensive tuning to determine the optimal configuration of crucial hyperparameters, specifically the number of convolutional kernels and the learning rate, in order to achieve the best performance. The effectiveness of the proposed model was evaluated using the NSL-KDD dataset, demonstrating its strong capability in detecting DDoS attacks.

The primary challenge addressed in this study is improving detection time and accuracy compared to the state-of-the-art methods above. To achieve this, we propose an improved GA to optimize the hyperparameters of the ELM model. Our optimized ELM model efficiently and effectively detects DDoS patterns in cloud computing, with a low rate of false positives.

ELM and GA Background

In this section, we provide an overview of GA steps, the ELM scheme, and the process of its output calculation.

ELM

An ELM (Wang *et al.*, 2022b) is a type of neural network with a single hidden layer, where the weights connecting the input layer to the hidden layer, as well as the biases, are randomly assigned and fixed, while the weights linking the hidden to the output layer are calculated analytically by Moore-Penrose inverse. These randomly set parameters do not require adjustment during training, simplifying the learning process and making the training time of the model very fast. Figure (1). presents the ELM scheme, where a, b, and c are the number of features in samples, the number of neurons in a hidden layer, and the number of neurons in an output layer, respectively.

$w_{i,n}$ is the weight connecting the i_{th} input to the n_{th} hidden neuron, the set of all weights is represented by *the* W matrix. $\beta_{n,j}$ represents the weight connecting the n_{th} hidden neuron to j_{th} output neuron and the set of all these weights is represented by β matrix. b_i is the bias of the i_{th} hidden neuron (i = 1-b):

$$W = \begin{bmatrix} w_{1,1} & . & w_{1,b} \\ . & . & . \\ w_{a,1} & . & w_{a,b} \end{bmatrix} \beta = \begin{bmatrix} \beta_{1,1} & . & \beta_{1,c} \\ . & . & . \\ \beta_{b,1} & . & \beta_{b,c} \end{bmatrix}$$

Suppose, the ELM in Fig. (1) has an activation function f with N training samples of the form: $S = (x_i,y_i)$, $x_i = (x_{i1},x_{i2},\ldots, x_{ia})^T \in R^N$, $y_i = (y_{i1},y_{i2},\ldots,y_{ic})^T \in R^c$, where x_i refers the input value and y_i represents the target, the output o_i f an ELM with b hidden neurons can be expressed in Eq. (1) as:

$$\sum_{i=1}^{b} \beta_i H_i = o_j, j = 1,..,c \quad (1)$$

where, $H_i = f(w_i x_k + b_i)$, k = 1,.....,a.

The training aims to reduce the error between the target and the output of ELM. The most frequently used objective function is Mean Squared Error (MSE), which is given by Eq. (2):

$$MSE = \sum (y_{ij} - o_{ij})^2, j = 1,..,c \quad (2)$$

ELM can approximate the output of all training samples to the target with Eq. (3), which is called the universal approximation capability:

$$\sum \|y_{ij} - o_{ij}\| = 0 \quad (3)$$

So, there must be a set of w_i, b_i, and β_i that suffices the Eq. (4):

$$\sum_{i=1}^{b} \beta_i H_i = y_j \quad (4)$$

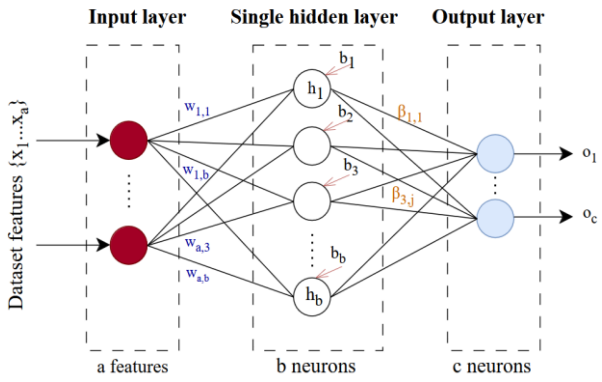

Fig. 1: ELM architecture

The formula above can be abbreviated as: $H\beta = Y$ Therefore, training ELM is finding the best w_i and b_i. These parameters are initialized randomly and independently of the input data and then the output weights are calculated by Eq. (5):

$$\beta = H^{+}Y \tag{5}$$

where, H^{+} is the Moore-Pensore inverse of matrix H.

Genetic Algorithm

Genetic Algorithm (Mitchell, 2016) is a technique of soft computing, that uses the laws of selection and evolution, it is essentially used to discover the optimal solution to machine learning problems such as feature selection in the dataset, choice of algorithm's hyperparameters, etc. GA iteratively evolves a population of solutions to an optimization problem through processes like selection, crossover, and mutation. The following mathematical representation of the key components of a genetic algorithm:

- Population Generation
  A population of potential solutions is initialized, typically at random. Let the population be represented as $P(t) = (x_1(t),x_2(t),\ldots,x_N(t))$ Where: $P(t)$ is the population at generation t, $x_i(t)$ is the i_{th} individual in the population at generation t, N is the population size.
- Fitness function
  A fitness function $f(x)$ is used to evaluate each individual in the population, determining how good a solution is for the problem. The fitness of individual $x_i(t)$ is represented by Eq (6):

$$f(x_i(t)) = \frac{true\ classified\ instances}{true + false\ instances} \tag{6}$$

The goal is typically to maximize the fitness function over populations:

- Selection
  Individuals are selected based on their fitness scores to reproduce and form the next generation. Higher-fitness individuals have a higher chance of being selected. The probability $p(x_i)$ of selecting individual x_i is often proportional to its fitness and represented by Eq (7):

$$p(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)} \tag{7}$$

where, $f(x_j)$ is the fitness of individual x_j.

- Crossover
  Crossover combines two parent solutions to create offspring. If two parents p_1 and p_2 are selected, their offspring o_1 and o_2 can be generated by a crossover operation. For example, in single-point crossover:

$$p_1 = (p_{11},p_{12},\ldots,p_{1k},\ldots,p_{1n})$$
$$p_2 = (p_{21},p_{22},\ldots,p_{2k},\ldots,p_{2n})$$

After crossover at point k:

$$o_1 = (p_{11},p_{12},\ldots,p_{1k},p_{2(k+1)},\ldots,p_{2n})$$
$$o_2 = (p_{21},p_{22},\ldots,p_{2k},p_{1(k+1)}\ldots,p_{1n})$$

In two-point crossover:

$$p_1 = (p_{11},\ldots,p_{1k},\ldots,p_{1k'},\ldots,p_{1n})$$
$$p_2 = (p_{21},\ldots,p_{2k},\ldots,p_{2k'},\ldots,p_{2n})$$

After crossover at points k and k':

$$o_1 = (p_{11},p_{1k},p_2,p_{2(k+1)},\ldots,p_{2(k'-1)},p_{1k'},\ldots,p_{1n}$$
$$o_2 = (p_{21},p_{2k},p_1,p_{1(k+1)},\ldots,p_{1(k'-1)},p_{2k'},\ldots,p_{2n}$$

This creates two new offspring by exchanging segments of the parents.

- Mutation
  Mutation introduces random changes to an individual to maintain genetic diversity. For a mutation rate μ, some genes x_{ij} of an individual x_i are mutated.

In a genetic algorithm, the steps below are repeated until an optimal solution is obtained or a stopping criterion is met.

Our Proposed Model: GA-OELM

The flowchart in Fig. (2). illustrates the functioning of our proposed DDoS detection system, which consists of three main modules: data preparation, ELM training, and online DDoS attack detection.

Data Preparation

Data preparation is the stage where the data sample is prepared for a classifier to improve its accuracy. During the training phase, this involves two key steps: Data preprocessing and data splitting (Alexandropoulos *et al*., 2019):

- Data preprocessing involves several essential tasks: First, cleaning the data to eliminate missing values and outliers; second, converting categorical data into numeric format; third, conducting feature selection to identify the most relevant features from the dataset using statistical algorithm as Anova; fourth, normalizing the data to ensure that all features contribute equally to the model by scaling all values to the range of [0, 1] using the following Eq. (8); and finally, applying target encoding to designate benign traffic as 0 and malicious traffic as 1:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{8}$$

where, x_{min} and x_{max} are the minimum and maximum values of the given feature x:

– Data splitting involves dividing the dataset into two subsets: A training subset, which is used to build the model, and a testing subset, which is utilized to assess the model's performance and verify its accuracy.

During the detection phase, incoming and internal network traffic is captured and then organized into groups. These groups are passed to the preprocessing machine to extract features as those used in the training phase and then to prepare them the same way to be ready for use by the GA-OELM classifier.

ELM Training

Our proposed GA-OELM is a supervised model for DDoS attack detection that requires training with labeled samples. These samples are already preprocessed in the data preparation stage and they are of the form $[x_i,y_i]$. Here, $x_i = [x_{i1}, ..., x_{ia}]$ the features for i^{th} training sample and y_i is the target, it is equal to 0 in case of normal traffic and 1 if not. The training phase uses a genetic algorithm to find the values of the weight matrix (w) and hidden biases, which allow the ELM classifier to achieve high detection accuracy. At first, a population of chromosome vectors is created. For each chromosome, the fitness function and β matrix are calculated. Then based on fitness results, individuals are selected from a population to perform mutation and crossover operations:

- Representation of chromosome
  The chromosome size is linked to the architecture of ELM, taking an example from Fig. (3). b neurons are used in the hidden layer, a is the number of features in each sample. Therefore, the length L of the chromosome contains all the weights connecting the input to the hidden layer as well as the biases $L= (a*b) + b$. A random initialization of N chromosomes is carried out. The training process uses each chromosome to train the ELM model. It divides a chromosome into two vectors, the first one is reshaped to the matrix of $a*b$ dimension and used by input-hidden connection weights, and the second vector is used for biases. Then ELM model calculates the corresponding matrix β as we saw in Eq. (5)
- Fitness function and selection
  After training the ELM model, it is used to predict the output of testing samples. For each chromosome, the fitness function is computed. Then parents are selected by roulette wheel technique to produce the new children, who will replace their parents in the new population
- Crossover and mutation
  At the crossover step, a random number C_r is generated and based on its value the crossover process is or not performed. One of two types of crossovers is selected either a single-point or two-point crossover. Whether the crossover did not perform, a new child with random values is generated to explore the searching space looking for potentially better value of weights and biases. In the mutation operation, the random gene of the selected chromosome is chosen and replaced by a random float from a specific range. These processes continue with the next generation until the stopping criteria are met. Ultimately, the best vector in the population represents the optimal values for weights and biases of the hidden layer. Algorithm 1 represents the training phase of the proposed system

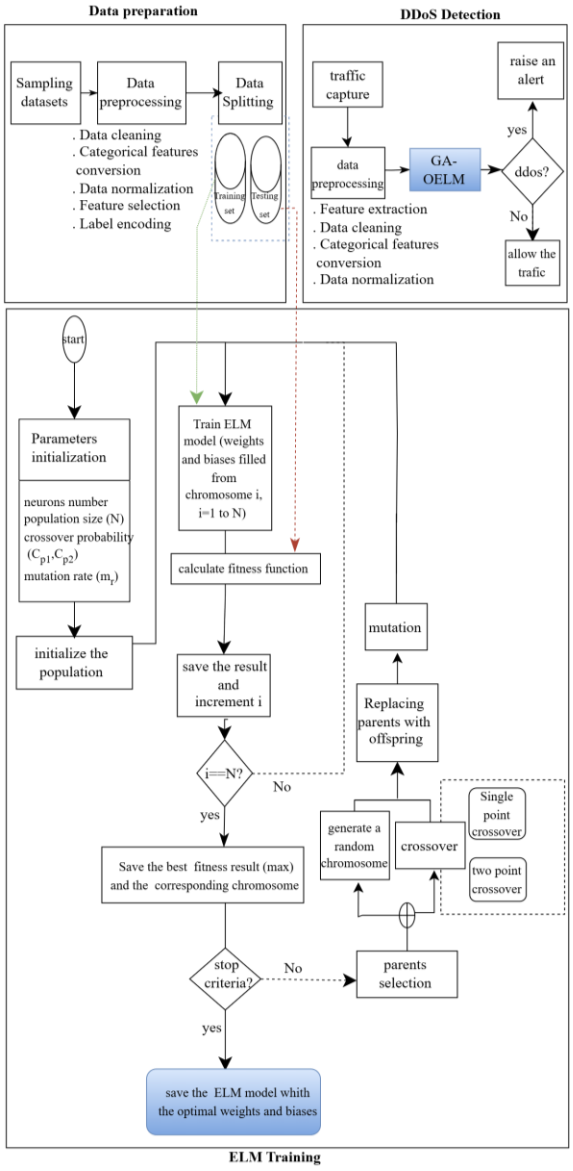

Fig. 2: Flowchart of the proposed system

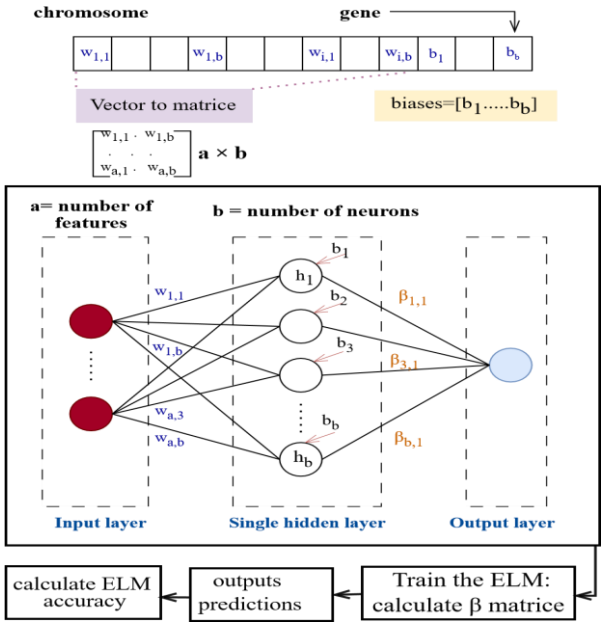

Fig. 3: ELM training

DDoS Attack Detection

At this stage, the traffic captured from the target cloud network is sent to the DDoS detection system. The process begins with the data preparation module, which transforms the traffic into samples in the form $x_i = [x_1,...,x_a]$. These samples are then fed into the trained GA-OELM model to compute the output o_i. The model classifies each sample based on its prediction. Specifically, if $o_i = 0$, the sample is identified as normal traffic; conversely, it is classified as an attack, triggering a notification to the cloud manager. The detection process of our system is outlined in Algorithm 2.

Results

The Used Datasets

The proposed GA-OELM is applied on three widely used datasets, CICDDoS2019, which contains DDoS attacks and benign traffic. A CICFlowMeter is used to process the network traffic. There are 18 CSV files saved for 2 days, the dataset has eighty-eight features, contains more than 5 million records of DDoS attacks and benign traffic, and has 12 types of DDoS attacks (Sharafaldin *et al*., 2019).

NSL-KDD dataset, which has forty-one features, the training subset KDDTrain+.txt includes 125,973 samples and two test subsets KDDTest-21.txt and KDD Test+ .txt, they include 11,850 and 22,544 samples, respectively. The attacks of this dataset are categorized into four categories, DOS, User to Root (U2R), Remote to Local (R2L), and Probe (Tavallaee *et al*., 2009).

Algorithm 1: ELM training

1. Initialize x = dataset[features], y = dataset[label], L = chromosome length, G = 100, N = 10, $M_r = 0.05$, $S_p = 0.75$, $C_{p1} = 0.4$, $C_{p2} = 0.7$
2. Randomly initialize a population of N vectors of length L
3. For Generation ≤ G do:
4. For each individual in the population do:
5. Convert chromosome to weights matrix of (a*b) dimension and bias vector b
6. Calculate H = f (x*w+ b)
7. Calculate = H^+Y
8. Calculate fitness using the accuracy metric
9. End For
10. Designate the individual with the best fitness as D_{best}
11. For iteration in t = round($N*S_p$)/2 (crossover process is repeated t times):
12. Select two parents from the population
13. IF randnum≤Cp1: Single point crossover is applied
14. ELSEIF C_{p1} < randnum≤C_{p2}: Two-point crossover is applied
15. ELSE: generate a new child randomly
16. ENDIF
17. Replacing the selected parents with new offspring
18. End For
19. IF randnum ≤ M_r: mutation is selected
20. Pass the new population to the next generation
21. Calculate the fitness function for each solution
22. Record individual with the best fitness D_{best}
23. End For
24. Return the trained GA-OELM model (trained with D_{best})

Algorithm 2: DDoS attack detection

1. For each sample x_i do:
2. Apply the sample to the optimized ELM
3. Calculate the output o_i
4. IF o_i == 1:
5. Attack detected
6. Generate alert
7. End IF
8. End For

UNSW-NB15 dataset, which includes 9 types of attacks named, DoS, Reconnaissance, Analysis, Exploits, Backdoor, Fuzzers, Shellcode, Generic, and Worms. It has a training subset and a testing subset with 175,341 samples and 82,332 samples, respectively. The dataset has 2,540,044 samples and forty-eight features (Moustafa and Slay, 2015).

CICDDoS2019, NSL-KDD, and UNSW-NB15 datasets have significant limitations, including synthetic or outdated attacks that may not represent modern threats or real-world complexity, class imbalances, and the need for extensive preprocessing. However, they are valuable for academic research due to their accessibility, historical

importance, and ability to provide standardized benchmarks, which facilitate model comparisons and serve as useful starting points for developing new DDoS detection systems.

Performance Metrics

Several metrics are used to assess the performance of the DDoS detection model; the following are applied to our model Sensitivity (Sen), Accuracy (Acc), Specificity (Sp), and False Positive Rate (FPR). The used metrics are presented in Eqs. (9-12) as the following:

$$Sen = \frac{TP}{TP+FN} \quad (9)$$

$$Acc = \frac{TN+TP}{TN+FN+TP+FP} \quad (10)$$

$$Sp = \frac{TN}{TN+FN} \quad (11)$$

$$FPR = \frac{FP}{TN+FP} \quad (12)$$

where, True Positive (TP) is an instance, which is correctly detected as an actual intrusion, False Positive (FP) is an instance, which is a legitimate network activity and identified as a DDoS attack. True Negative (TN) is an instance, which is correctly identified normal network activity as not being a DDoS attack. False Negative (FN) is an instance when the detection system fails to identify an actual DDoS attack.

Training Setup

In this experiment, we used a machine with an i5 CPU 1.70 GHz, 16 Go of RAM, and Python 3.9. We employed the training set and testing set reserved for both UNSW-NB15 and NSL-KDD and only a part from the CICDDoS2019 dataset to avoid the high computational cost. The used subsets are split into a training set with 80% and a testing set with 20% to compare results (Alqarni, 2022). Table (1) displays the used part of the CICDDoS2019 dataset.

For the proposed GA-OELM, we prepared the datasets by consolidating all attack-type labels into a single "attack" label. We transformed nominal values into numeric formats, normalized the data, and encoded the labels. In this experiment, we retained all features from the UNSW-NB15 and NSL-KDD datasets to ensure that our results are comparable to those of previous researchers mentioned in the literature review. For CICDDoS2019, the Anova technique is used from the scikit-learn library to select the most relevant feature.

In this article, a comprehensive performance analysis of our model is done. For that, in the ELM hidden layer, we varied neuron numbers from 10-50 in the multiple of 10 and for each number of neurons we make 100 generations, keeping the number of individuals in the population constant. The parameters in Table (2) are used for training.

Results

Figure (4) displays the testing accuracy for different numbers of hidden neurons. We observe that the accuracy of GA-OELM improves consistently across all datasets as the number of neurons increases. We begin training the model with 10 neurons for 100 generations, gradually increasing the number of neurons up to 50 in increments of 10. We observe an improvement of nearly 2% in accuracy when comparing 10 and 50 neurons across most datasets. The accuracy tends to increase significantly with more neurons. Ultimately, we opt for 50 neurons in the hidden layer to maintain manageable computational demands. Additionally, this accuracy is competitive compared to several state-of-the-art models.

Figure (5) illustrates the testing accuracy curve of our GA-OELM model (with 50 neurons) across all datasets over the number of generations. Notably, accuracy improves as the number of generations increases, with most datasets showing a 10-20% increase in accuracy from the first to the 50th generation. After 50 generations, the improvement diminishes to just 0.02%. The KDD Test+ dataset stands out, showing a significant accuracy increase after the 70th generation, rising from 89.79 to 93.55% by the 100th generation. Although other datasets might continue to improve beyond 100 generations, we ended the experiment due to a lack of significant progress over the last 40 generations.

Table 1: The used CICDDoS2019 subset

Total records	Benign records	DDoS records
32000	8450	23550

Table 2: Training parameters

Parameter name	Value
Population size: N	10
Hidden neuron: B	10-50 (step:10)
Mutation rate: M_r	5%
Selection probability: S_p	75%
Crossover type: C_{p1}, C_{p2}	C_{p1}=0.4, C_{p2}=0.7

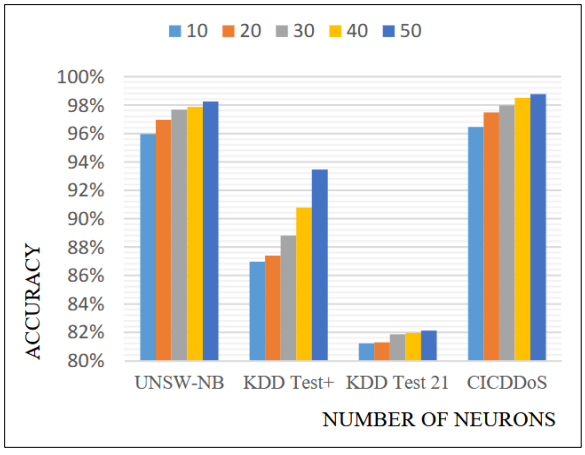

Fig. 4: Testing accuracy of different numbers of neurons

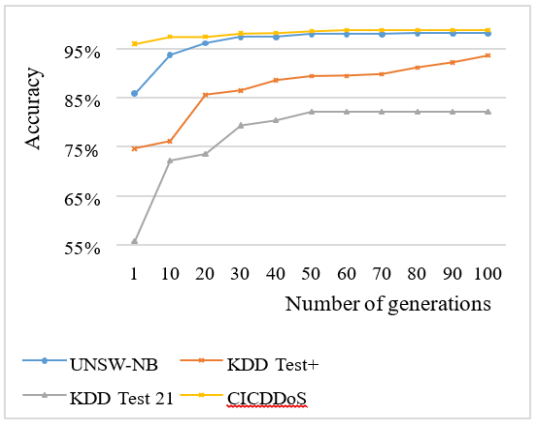

Fig. 5: Accuracy curve over generations

Table (3) presents the performance metrics (accuracy, False Positive Rate (FPR), sensitivity, and specificity) for the ELM model optimized using PSO, ACO, Gradient Descent, and an improved GA. The model is evaluated with 50 hidden neurons, across 100 generations and using the same subsets of datasets. GA-OELM demonstrates the highest accuracy across all datasets, peaking at 99.11% on NSLKDD and 98.81% on CICDDOS. PSO-ELM also performs well, particularly on NSLKDD with 97.29% and CICDDOS with 97.25%. As shown in Fig. (6). GA-OELM maintains consistently low false positive rates (1.41-1.52%) across all datasets, effectively minimizing false alarms and reducing the unnecessary blocking of normal traffic, while ACO-ELM shows significantly higher FPRs, particularly at 55.51% on CICDDOS, indicating challenges in accurate classification. GA-OELM demonstrates the highest sensitivity for CICDDOS at 99.50%, effectively identifying positive instances (attacks), while PSO-ELM shows strong sensitivity at 97.71% on UNSW-BN. In terms of specificity, GA-OELM maintains high rates (98-99%), accurately identifying negative instances, whereas ACO-ELM struggles with low specificity at 56.90% on CICDDOS, indicating a tendency to misclassify normal traffic as an attack. Gradient Descent-ELM performs moderately, with decent accuracy but lower sensitivity and specificity compared to GA-OELM and PSO-ELM. The improved GA outperforms PSO, ACO, and gradient descent in selecting the optimal hyperparameter for the ELM, resulting in better overall performance.

The performance of the proposed GA-OELM model is compared with several widely used machine learning models, including Support Vector Machines (SVM), Naive Bayes (NB), Artificial Neural Networks (ANN), and classic Extreme Learning Machines (ELM). The configuration details are as follows: The classic ELM model consists of 100 neurons in the hidden layer, using a sigmoid activation function. The ANN model features a single hidden layer with 100 neurons, employing the Re LU activation function for the hidden layer and the sigmoid function for the output layer. For the SVM, we utilize the 'SVC 'function from Scikit-learn with a polynomial kernel and a maximum of 50 iterations. The Naive Bayes model uses the 'Gaussian NB' function from Scikit-learn with its default parameters.

Table (4) presents the highest accuracy values obtained from 10 experiments for training the models on each dataset. Table (4) highlights the significant improvements in accuracy, sensitivity, and specificity offered by the proposed GA-OELM model compared to the four benchmark systems across all datasets. Notably, the SVM model produced poor classification results, achieving only 52 and 50.73% accuracy for the KDD Test+ and KDD Test21 datasets, respectively. In contrast, our GA-OELM model demonstrates an impressive improvement of approximately 40%. Furthermore, when trained using the ANN technique, the CIC DDoS 2019 dataset and UNSW-NB 15 recorded low accuracies of 67 and 81%, respectively, our GA-OELM model also outperformed these results with a significant improvement of over 15%. GA-OELM model demonstrates superior performance compared to the other models, consistently achieving higher accuracy, sensitivity, and specificity across all datasets.

Table 3: Performance metrics of ELM Model Optimized by different algorithms

Model	Dataset	Accuracy%	FPR %	Sensitivity%	Specificity%
PSO-ELM	NSLKDD	97.29	1.56	95.83	98.43
	CICDDOS	97.25	0.442	90.60	99.55
	UNSW-BN	90.18	23.08	97.71	76.91
ACO-ELM	NSLKDD	79.29	35.30	95.83	64.69
	CICDDOS	63.52	55.51	78.48	56.90
	UNSW-BN	74.3	1.472	60.52	98.52
Gradient Descent-ELM	NSLKDD	96	2.51	96.80	97.48
	CICDDOS	95.31	1.57	86.36	98.42
	UNSW-BN	89	27.92	98.87	72
GA-OELM	NSLKDD	99.11	1.52	99.20	99
	CICDDOS	98.81	1.41	99.50	98.58
	UNSW-BN	98.16	1.52	98.25	98.47

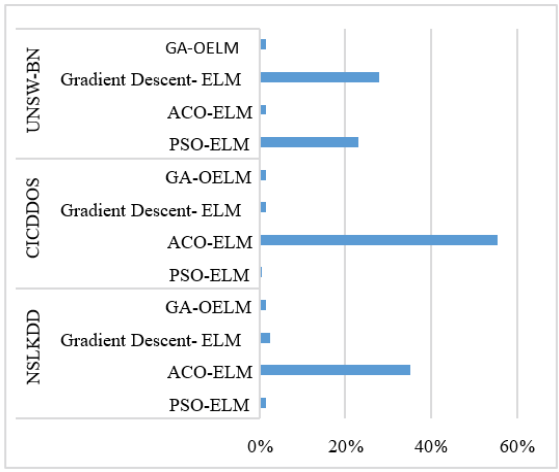

Fig. 6: FPR of ELM model optimized by various algorithms

Table 4: Performance comparison with the widely used techniques

Model	Dataset	Accuracy %	Sensitivity %	Specificity %
ANN	KDD TEST+	64.29	55	90
	KDD TEST21	70.28	30.45	87
	CICDDOS	67	41.21	84.65
	UNSW-BN	81	15	97.7
ELM	KDD TEST+	74.32	63.79	90.33
	KDD TEST 21	55.21	23.81	87.47
	CICDDOS	96.45	98.11	85.61
	UNSW-BN	89.53	29.58	99.14
SVM	KDD TEST+	52	52	52.62
	KDD TEST 21	50.73	50	54.70
	CICDDOS	80.29	98	79
	UNSW-BN	90.33	4	95
NB	KDD TEST+	80.70	72.94	90.98
	KDD TEST 21	80.69	72.30	88
	CICDDOS	97.51	98	96.96
	UNSW-BN	91.37	32.47	98.58
GA-OELM	KDD TEST+	93.55	91.58	95.10
	KDD TEST 21	82.16	79.30	88.38
	CICDDOS	98.81	99.50	98.58
	UNSW-BN	98.16	98.25	98.47

The performance of our GA-OELM model is also compared in Table (5) to the state-of-the-art techniques. The GA-OELM model consistently outperforms many existing methods. On the NSLKDD dataset, GA-OELM achieves the highest accuracy with 99.2% and a strong balance between sensitivity 99.09% and specificity 99.39%, slightly exceeding models used by Kushwah and Ranga (2020); Velliangiri *et al*. (2021); Lin *et al*. (2023); Arunadevi and Sathya (2022); Hussein Hadi (2024). Ali and Kushwah (2019) used an ELM model with 613 neurons to achieve 99.10% accuracy. In contrast, our model, with only 400 neurons, achieved 99.2% accuracy. This demonstrates that we improved accuracy while significantly reducing detection time. On the CICDDoS dataset, GA-OELM achieves the highest accuracy of 98.81% and a sensitivity of 98.25%, demonstrating its strong ability to detect true positives accurately. Additionally, with a high specificity of 98.47%, the model effectively distinguishes normal traffic from DDoS attacks. Our GA-OELM shows a small improvement in terms of accuracy compared to Alqarni (2022) but the detection time of the ELM model is still much shorter than that of the majority voting technique used by the authors. For KDD TEST+ and KDDTEST21 datasets, GA-OELM also outperforms previous models, achieving a higher accuracy of 93.55% for KDD TEST+. Finally, on the UNSW-BN dataset, GA-OELM demonstrates strong overall performance with 98.16% accuracy, 98.25% sensitivity, and 98.47% specificity, showcasing its robustness in various contexts. Overall, the GA-OELM model consistently achieves high accuracy across the datasets and generally maintains a good trade-off between sensitivity and specificity, outperforming several existing methods in most cases.

Our proposed GA-OELM system effectively identifies DDoS attacks and outperforms systems based on ANN, SVM, and NB, as well as other state-of-the-art approaches. It shows that using both single-point and two-point crossover, along with random offspring initialization, is more effective than classical crossover methods. As shown in Table 6, the main drawback is the longer training time, typical of genetic algorithms. However, our system performs very quickly during the testing phase, making it highly suitable for detecting attacks in cloud computing environments.

Application to Real-World Scenario

A Cloud Service Provider (CSP) hosts numerous critical applications for businesses and users worldwide. The CSP is a prime target for DDoS attacks, which can disrupt services and cause significant financial damage. To prevent this, the CSP needs to deploy our proposed DDoS detection system.

Table 5: Performance comparison with the state of arts

Model	Dataset	Accuracy %	Sensitivity %	Specificity %
Kushwah and Ranga (2020)	Partial NSLKDD	99.18	99.50	98.86
Arunadevi and Sathya (2022)		98.72	99.75	98.61
Lin *et al*. (2023)		96.53	-	-
Hussein Hadi (2024)		97.27	97.9	97.27
Velliangiri *et al*. (2021)		99.10	99.19	98.96
GA-OELM		99.2	99.09	99.39
Alqarni (2022)	CICDDOS	98.02	97.45	98.65
Kumar *et al*. (2023)		98	97	-
GA-OELM		98.81	98.25	98.6
Kushwah and Ranga (2021)	KDD TEST+	86.80	79.30	95.90
	KDD TEST 21	73.00	59.50	90.20
	UNSW-BN	89.17	80.73	99.58
GA-OELM	KDD TEST+	93.55	91.58	95.10
	KDD TEST21	82.16	79.30	88.38
	UNSW-BN	98.16	98.25	98.47

Table 6: Training and testing times

	Dataset	
Training time (sec) per ELM instance	KDDTrain+	0.6398
	USWN	0.7568
	CICDDoS	0.1010
Testing time (sec)	KDDTest	0.0571
	USWN	0.0145
	CICDDoS	0.1505

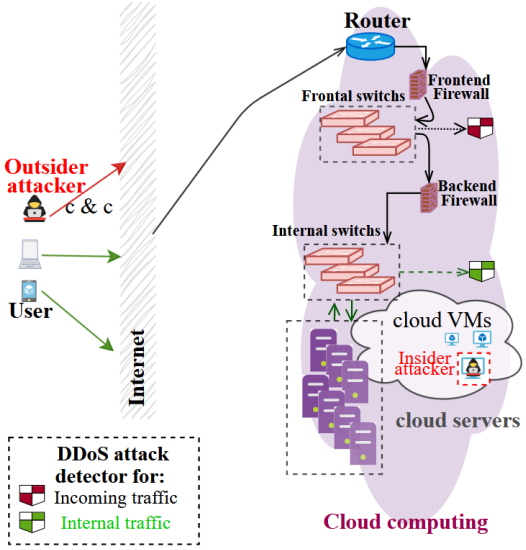

Fig. 7: The proposed placement of DDoS detection system

Scenario 1: A DDoS attack originating from outside the cloud. As shown in Fig. (7), our DDoS attack detection system is positioned at the network's front end to monitor all incoming traffic before it enters the CSP platform. Scenario 2: A DDoS attack originating from inside the cloud. As illustrated in Fig. (7), our DDoS attack detection system is strategically placed between the internal switches and the hypervisor to detect any DDoS traffic originating from internal attackers aiming to compromise the CSP resources. The CSP needs to place a DDoS detector at different places to continuously monitor all incoming traffic to its cloud infrastructure, in addition to critical internal traffic. This captured traffic is then collected and sent to the DDoS detection system. Where it first undergoes preprocessing to prepare the data for analysis. Once processed, the data is passed to the detection system, which computes the output using the trained GA-OELM model. Once an attack is detected, the system sends a notification to the cloud managers and potentially activates automated countermeasures, such as traffic filtering or rerouting, to mitigate the impact of the attack. The cloud managers receive a notification and proceed to investigate whether the alert is a true positive. If it is confirmed as a true positive, the countermeasures remain active. If it is a false positive, the managers allow the traffic and plan for future training by using the historical and new data to compute a new baseline, thereby, optimizing the GA-OELM performance.

Conclusion

Cloud computing provides resources as services over the Internet. Therefore, their availability is of utmost importance. In the present paper, the proposed GA-OELM is used for DDoS attack detection in cloud environments. The model is optimized by using a genetic algorithm with two types of crossovers in addition to enriching frequently the population by random children to further explore the searching space. The GA-OELM reaches an accuracy of 98.81, 93.55, 82.16 and 98.16% for CICDDoS2019, KDD Test+, KDDTest21, and USWN-BN, respectively, which is better compared to ANN, SVM, NB and some stat of art techniques.

Acknowledgment

We express our profound gratitude to FST, ISA laboratory for providing the necessary facilities for conducting this research.

Funding Information

This research was conducted independently by the authors without external financial support or assistance.

Author's Contributions

Meryem Ec-Sabery: Participated in all experiments, coding, building the trained models. She also evaluated the results and made significant contributions to the writing of the manuscript.

Rachid Ben Abbou: Designed the research plan, revised and edited the manuscript.

Fatiha Mrabti: Provided essential guidance, revised and edited the manuscript.

Adil Ben Abbou and Abdelali Boushaba: Reviewed the manuscript and provided insightful, constructive feedback.

Ethics

The present study represents an original research effort. The corresponding author confirms that all co-authors have reviewed and approved the manuscript, with no ethical issues raised.

Reference

Abu Al-Haija, Q., Altamimi, S., & AlWadi, M. (2024). Analysis of Extreme Learning Machines (ELMs) for Intelligent Intrusion Detection Systems: A Survey. *Expert Systems with Applications*, *253*, 124317. https://doi.org/10.1016/j.eswa.2024.124317

Agrawal, N., & Tapaswi, S. (2019). Defense Mechanisms Against DDoS Attacks in a Cloud Computing Environment: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, *21*(4), 3769–3795. https://doi.org/10.1109/comst.2019.2934468

Alexandropoulos, S.-A. N., Kotsiantis, S. B., & Vrahatis, M. N. (2019). Data Preprocessing in Predictive Data Mining. *The Knowledge Engineering Review*, *34*, e1. https://doi.org/10.1017/s026988891800036x

Ali, S. T., & Kushwah, G. S. (2019). Distributed Denial of Service Attacks Detection in Cloud Computing Using Extreme Learning Machine. *International Journal of Communication Networks and Distributed Systems*, *23*(3), 328–351. https://doi.org/10.1504/ijcnds.2019.10022365

Alqarni, A. A. (2022). Majority Vote-Based Ensemble Approach for Distributed Denial of Service Attack Detection in Cloud Computing. *Journal of Cyber Security and Mobility*, *11*(2), 265–278. https://doi.org/10.13052/jcsm2245-1439.1126

Alsoufi, M. A., Siraj, M. M., Ghaleb, F. A., Al-Razgan, M., Al-Asaly, M. S., Alfakih, T., & Saeed, F. (2024). Anomaly-Based Intrusion Detection Model Using Deep Learning for IoT Networks. *Computer Modeling in Engineering & Sciences*, *141*(1), 823–845. https://doi.org/10.32604/cmes.2024.052112

Arunadevi, M., & Sathya, V. (2022). *Optimized Back Propagation Neural Network for Ddos Attack Detection in the Cloud Environment*.

Balarezo, J. F., Wang, S., Chavez, K. G., Al-Hourani, A., & Kandeepan, S. (2022). A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks. *Engineering Science and Technology, an International Journal*, *31*, 101065. https://doi.org/10.1016/j.jestch.2021.09.011

Canfora, G., Di Sorbo, A., Mercaldo, F., & Visaggio, C. A. (2015). Obfuscation Techniques against Signature-Based Detection: A Case Study. *2015 Mobile Systems Technologies Workshop (MST)*, 21–26. https://doi.org/10.1109/mst.2015.8

Cloud, H. (2011). The Nist Definition of Cloud Computing. *National Institute of Science and Technology, Special Publication*, *800*(2011), 145.

Dorigo, M., & Stutzle, T. (2019). Ant Colony Optimization: Overview and Recent Advances. In M. Gendreau & J. Y. Potvin (Eds.), *International Series in Operations Research & Management Science* (Vol. 272, pp. 311–351). Springer. https://doi.org/10.1007/978-3-319-91086-4_10

Efendi, R., Wahyono, T., & Widiasari, I. R. (2024). DBSCAN SMOTE LSTM: Effective Strategies for Distributed Denial of Service Detection in Imbalanced Network Environments. *Big Data and Cognitive Computing*, *8*(9), 118. https://doi.org/10.3390/bdcc8090118

Hubballi, N., & Suryanarayanan, V. (2014). False Alarm Minimization Techniques in Signature-Based Intrusion Detection Systems: A Survey. *Computer Communications*, *49*, 1–17. https://doi.org/10.1016/j.comcom.2014.04.012

Hussein Hadi, T. (2024). Deep Learning-Based DDoS Detection in Network Traffic Data. *International Journal of Electrical and Computer Engineering Systems*, *15*(5), 407–414. https://doi.org/10.32985/ijeces.15.5.3

Jon-Michael, C. B., A., G., & Greg, J. (2020). *Top Threats to Cloud Computing the Egregious 11*.

Karaboga, D., & Basturk, B. (2007). A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Kumar, D., Pateriya, R. K., Gupta, R. K., Dehalwar, V., & Sharma, A. (2023). DDoS Detection Using Deep Learning. *Procedia Computer Science*, *218*, 2420–2429. https://doi.org/10.1016/j.procs.2023.01.217

Kumar, S., Dwivedi, M., Kumar, M., & Gill, S. S. (2024). A Comprehensive Review of Vulnerabilities and AI-Enabled Defense against DDoS Attacks for Securing Cloud Services. *Computer Science Review*, *53*, 100661. https://doi.org/10.1016/j.cosrev.2024.100661

Kushwah, G. S., & Ranga, V. (2020). Voting Extreme Learning Machine Based Distributed Denial of Service Attack Detection in Cloud Computing. *Journal of Information Security and Applications*, *53*, 102532. https://doi.org/10.1016/j.jisa.2020.102532

Kushwah, G. S., & Ranga, V. (2021). Optimized Extreme Learning Machine for Detecting Ddos Attacks in Cloud Computing. *Computers & Security*, *105*, 102260. https://doi.org/10.1016/j.cose.2021.102260

Lata, S., & Singh, D. (2022). Intrusion Detection System in Cloud Environment: Literature Survey & Future Research Directions. *International Journal of Information Management Data Insights*, *2*(2), 100134. https://doi.org/10.1016/j.jjimei.2022.100134

Lin, H., Xue, Q., Feng, J., & Bai, D. (2023). Internet of Things Intrusion Detection Model and Algorithm Based on Cloud Computing and Multi-Feature Extraction Extreme Learning Machine. *Digital Communications and Networks*, *9*(1), 111–124. https://doi.org/10.1016/j.dcan.2022.09.021

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). *2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6. https://doi.org/10.1109/milcis.2015.7348942

Mitchell, M. (2016). Melanie mitchell. ACM SIGEVOlution, 8(2): 4–4.

Ouhssini, M., Afdel, K., Akouhar, M., Agherrabi, E., & Abarda, A. (2024). Advancements in detecting, preventing and mitigating DDoS attacks in cloud environments: A comprehensive systematic review of state-of-the-art approaches. *Egyptian Informatics Journal*, *27*, 100517. https://doi.org/10.1016/j.eij.2024.100517

Selamat, A., Yusof, A. R., & Udzir, N. I. (2019). Systematic Literature Review and Taxonomy for DDoS Attack Detection and Prediction. *International Journal of Digital Enterprise Technology*, *1*(3), 292–315. https://doi.org/10.1504/ijdet.2019.10019068

Sharafaldin, I., Lashkari, A. H., Hakak, S., & Ghorbani, A. A. (2019). Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. *2019 International Carnahan Conference on Security Technology (ICCST)*, 1–8. https://doi.org/10.1109/ccst.2019.8888419

Somani, G., Gaur, M. S., Sanghi, D., Conti, M., & Buyya, R. (2017). DDoS attacks in cloud computing: Issues, taxonomy and future directions. *Computer Communications*, *107*, 30–48. https://doi.org/10.1016/j.comcom.2017.03.010

Songa, A. V., & Karri, G. R. (2024). An Integrated SDN Framework for Early Detection of DDoS Attacks in Cloud Computing. *Journal of Cloud Computing*, *13*(1), 64. https://doi.org/10.1186/s13677-024-00625-9

Syed, H. J., Gani, A., Ahmad, R. W., Khan, M. K., & Ahmed, A. I. A. (2017). Cloud monitoring: A review, taxonomy and open research issues. *Journal of Network and Computer Applications*, *98*, 11–26. https://doi.org/10.1016/j.jnca.2017.08.021

Shanmugapriya, D., Dhanya, C., Asha, S., Padmavathi, G. and Suthisini, D. N. P. (2024). Cloud insider threat detection using deep learning models. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 434–438. IEEE.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A Detailed Analysis of the KDD CUP 99 Data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6. https://doi.org/10.1109/cisda.2009.5356528

Velliangiri, S., Karthikeyan, P., & Vinoth Kumar, V. (2021). Detection of Distributed Denial of Service Attack in Cloud Computing Using the Optimization-Based Deep Networks. *Journal of Experimental & Theoretical Artificial Intelligence*, *33*(3), 405–424. https://doi.org/10.1080/0952813x.2020.1744196

Vibhute, A. D., & Nakum, V. (2024). Deep learning-based network anomaly detection and classification in an imbalanced cloud environment. *Procedia Computer Science*, *232*, 1636–1645. https://doi.org/10.1016/j.procs.2024.01.161

Wang, D., Tan, D., & Liu, L. (2018). Particle Swarm Optimization Algorithm: An Overview. *Soft Computing*, *22*(2), 387–408. https://doi.org/10.1007/s00500-016-2474-6

Wang, B., Hua, Q., Zhang, H., Tan, X., Nan, Y., Chen, R., & Shu, X. (2022a). Research on Anomaly Detection and Real-Time Reliability Evaluation with the Log of Cloud Platform. *Alexandria Engineering Journal*, *61*(9), 7183–7193. https://doi.org/10.1016/j.aej.2021.12.061

Wang, J., Lu, S., Wang, S.-H., & Zhang, Y.-D. (2022b). A review on extreme learning machine. *Multimedia Tools and Applications*, *81*(29), 41611–41660. https://doi.org/10.1007/s11042-021-11007-7

Wardana, A. A., Kołaczek, G., Warzyński, A., & Sukarno, P. (2024). Ensemble Averaging Deep Neural Network for Botnet Detection in Heterogeneous Internet of Things Devices. *Scientific Reports*, *14*(1), 3878. https://doi.org/10.1038/s41598-024-54438-6

Zhao, F., Li, H., Niu, K., Shi, J., & Song, R. (2024). Application of Deep Learning-Based Intrusion Detection System (IDS) in Network Anomaly Traffic Detection. *Applied and Computational Engineering*, *86*, 250–256. https://doi.org/10.54254/2755-2721/86/20241604

Zoppi, T., Gazzini, S., & Ceccarelli, A. (2024). Anomaly-Based Error and Intrusion Detection in Tabular Data: No DNN Outperforms Tree-Based Classifiers. *Future Generation Computer Systems*, *160*, 951–965. https://doi.org/10.1016/j.future.2024.06.051