International Journal of Cybersecurity Engineering and Innovation

ARTICLE INFO

Article History: Received: 24-01-2026, Revised: 20-02-2026, Accepted: 10-03-2026, Published: 14-03-2026
Corresponding author Email: abdal9991@gmail.com
DOI:

Vol. 2026 No.1

Accurate Network Intrusion Detection using a Feedforward Neural Network and Bee Colony Optimization Algorithm

Abbas Fadhil Kadhim[1], Abdulwahhab Essa Hamzah[1, 4*], Mahmood A. Al-Shareeda[2,3], Alaa Ibrahim Hussein[1], and Nurfarhana Mohamad Sapiee[4]

[1]*Department of Communications Engineering, Iraq University College, Basrah, Iraq*
[2]*Department of Electronic Technologies, Basra Technical Institute, Southern Technical University, Basra, Iraq.*
[3]*College of Engineering, Al-Ayen University, Thi-Qar, 64001, Iraq*
[4]*Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia (UKM), Bangi, 43600, Selangor, Malaysia*
**Corresponding Author: abdal9991@gmail.com*

Abstract

Intrusion detection in computer networks is essential for protecting sensitive information and maintaining network security. This paper proposes a hybrid intrusion detection approach using a feedforward multilayer perceptron (MLP) neural network optimized by the Artificial Bee Colony (ABC) algorithm instead of traditional gradient descent. The proposed model enables it to detect both known and novel attack patterns. The model is evaluated using the NSL-KDD dataset after preprocessing, normalization, and PCA-based feature reduction, and performance is assessed using accuracy, precision, recall, and F1-score.The optimization algorithm fine-tunes the network's weights and biases, leading to improved classification performance. The proposed model achieved an accuracy of 99.1%, surpassing other methods by an average of 3%, thus proving its effectiveness for secure and reliable intrusion detection.

Keywords: Intrusion detection, Machine learning, Multi-layer Perceptron Neural Network, Bee Colony Optimization Algorithm.

1. Introduction

In today's digital age, computers play a vital role across various aspects of daily life, supporting essential functions like electronic fund transfers, ATM operations, and the secure storage of medical, financial, and personal data. With such pervasive use, securing these systems against unauthorized access and malicious intent becomes a top priority. Mistakes, whether accidental or intentional, in information handling can lead to severe consequences, particularly as sophisticated cyber threats continue to evolve [1]. Modern threats not only stem from accidental errors but also include intentional actions by insiders, such as altering account details for personal gain or accessing sensitive data without authorization, leading to potential misuse of critical information [2, 3].

Network security, therefore, has emerged as one of the most critical components in safeguarding valuable resources against malicious attacks and preventing unauthorized access. Among the various tools designed to tackle these threats, Intrusion Detection Systems (IDS) stand out as essential for identifying unauthorized activities, misuse, or damage inflicted on networked systems [4]. IDS are generally divided into misuse detection and anomaly detection systems. Misuse detection identifies known patterns of malicious behavior, while anomaly detection compares current network activity with a baseline of normal behavior, flagging deviations as potential threats [5]. As cyber threats become increasingly sophisticated, achieving high accuracy and reliability in IDS has become a significant challenge. Machine learning offers a promising solution by improving detection capabilities and reducing false alarms, making it a highly effective approach for modern IDS [6].

Several studies have been conducted in relation to this issue. In [7], a novel method for identifying and preventing malicious network behavior is proposed to establish a secure network communication environment. The research addresses the challenge of network security, particularly in light of the increasing complexity and number of cyber-attacks in recent years. The study highlights the importance of network intrusion detection systems (NIDSs) as integral security measures to identify malicious activities. To tackle the issue of class imbalance commonly found in intrusion detection datasets, the proposed method utilizes a double-layer feature extraction and feature fusion technique (CNN-GRU-FF) that incorporates a modified focal loss function rather than traditional cross-entropy. The effectiveness of the model is evaluated using the NSL-KDD and UNSW-NB15 datasets, achieving impressive detection rates of 98.22% and 99.68%, respectively, while maintaining low false alarm rates. Furthermore, the performance of the proposed model is compared to seven baseline algorithms and existing methods in the literature, illustrating its superiority over state-of-the-art network intrusion detection techniques. In [8], a novel method is proposed to address the grave concerns regarding security and vulnerability in computer networks, particularly through the use of intrusion detection systems (IDS). This research highlights the challenges faced by current IDS devices, which tend to generate false alarms in response to normal user activities instead of effectively detecting novel attacks. To enhance detection accuracy, a hybrid approach that incorporates neural networks and correlation-based feature selection is introduced. The study employs the NSL-KDD standard dataset for experimental analysis of intrusion detection against current attacks. A unique hybrid crowd search analysis optimization with an artificial neural network (HCSAOANN) is proposed, demonstrating superior performance in terms of accuracy, precision, F1-Score, and recall. The HCSAOANN algorithm effectively merges crow search optimization (CSO) with an upgraded version of the crow search analysis method, facilitating exploration of the feature space to achieve optimal solutions. The proposed methodology achieves 98% accuracy and a 98% precision rate, surpassing previous techniques such as CSO-ANFIS by 2.2% and FC-ANN by 8.87%. Reference [9], a novel method is introduced for developing intelligent protection schemes in response to the increasing reliance of human systems on cyber infrastructures and the rise in cyber-attacks. The study emphasizes the demand for systems capable of detecting both existing and zero-day attacks in the complex and rapidly evolving cyberspace. While machine learning and deep learning models have shown effectiveness in this domain, the research aims to design and implement a Deep Neural Network (DNN) model for improved intrusion detection performance. To tackle data imbalance issues within the CICIDS 2017 dataset, techniques such as SMOTE and Random Sampling are utilized. The experiments are conducted within a single Jupyter notebook in the Google Colaboratory environment, using essential software libraries such as Seaborn, Pandas, Matplotlib, Keras, and TensorFlow. The model's performance is evaluated based on accuracy and loss metrics during training and validation phases. The findings reveal that the proposed deep learning model achieves a remarkable accuracy score of 99.68% and a loss of 0.0102, demonstrating its efficacy in predicting cyber-attacks.

Another reaserch [10], a novel long short-term memory (LSTM)-based near real-time multiclass network intrusion detection system (NIDS) is introduced to enhance the security and detection of network anomalies. The proposed method employs complex cloud CSE-CICIDSS2018 datasets to facilitate real-time intrusion detection. To optimize feature selection, a random forest algorithm is utilized for dimensionality reduction, ensuring the inclusion of only the most pertinent features in the LSTM model. The experimental findings reveal that the developed LSTM model achieved a remarkable testing accuracy of 99.66% with a loss of 0.12%, indicating its superior capability to detect network intrusions with high precision compared to previous designs.

In the reviewed articles, several challenges in intrusion detection using neural networks were identified. One major challenge is the high complexity of attacks and the continuous emergence of new intrusion methods, making it difficult for neural network-based models to recognize them. A high rate of false alarms is another issue that can lead to reduced efficiency and trust in detection systems. Additionally, the imbalance in training data, particularly the scarcity of intrusion samples compared to normal network behavior, is a key problem that can affect the accuracy of neural models. Moreover, processing large

volumes of network data and the need for real-time analysis impose a significant computational burden on deep learning models. These challenges highlight the need for improved and advanced methods to enhance the accuracy and effectiveness of intrusion detection using neural networks.

This paper presents a novel method for detecting network intrusion that leverages a feedforward neural network optimized through the Bee Colony Optimization (BCO) algorithm. The primary contribution of this research is the design and implementation of a feedforward neural network specifically tailored for network traffic analysis, enhanced by the BCO algorithm to optimize the model's parameters and improve its detection capabilities. The feedforward architecture enables efficient feature extraction and classification of network traffic, while the BCO algorithm, inspired by the foraging behavior of bees, facilitates an effective exploration of the parameter space, leading to improved convergence and accuracy. By integrating these two approaches, the proposed method not only enhances the performance of the intrusion detection system (IDS) under typical operating conditions but also increases its adaptability to varied network environments. This results in a more robust and reliable solution for detecting and mitigating network intrusions, ensuring better protection against potential threats.

2 . Research Background

2.1 Multilayer Perceptron Neural Network

Multilayer Perceptron (MLP) is an artificial neural network model consisting of multiple layers. Typically, an MLP consists of three main layers: the input layer, hidden layer(s), and the output layer.

Input Layer: Responsible for receiving features and input data for the problem. The number of neurons in this layer equals the number of input features. Features are directly input into the input layer without any transformation and are transferred to the hidden layer. *Hidden Layer(s):* These layers process information and extract hidden features from the data. The number and structure of hidden layers can vary, and there are usually multiple hidden layers. Each neuron in each hidden layer combines its inputs using weights and activation functions (such as sigmoid or tangent activation functions) and produces an output.

Output Layer: This layer produces the final result of the network. The number of neurons in this layer is usually determined by the number of classes or output features of the problem. Activation functions in this layer may be adjusted based on the type of problem, such as using the softmax function for classification tasks. In this neural network, information only moves forward and does not involve loops or cycles. When information moves from one layer to another, it undergoes weighted multiplication by values called weights and is summed with a constant term called bias. When reaching the output layer, these products of information are summed, and the output is obtained. Considering these explanations, the output of the neural network is evaluated as described in Equation 1:

$$O_j = f(\sum_{i=1}^{n} w_i X_i + b_j) \qquad (1)$$

In the above equation, O_j represents the output of the j-th neuron, X_i represents the i-th input, w_i is the weight of the i-th input, and b_j is the bias of the j-th neuron. f is the activation function, and various functions have been proposed for it, such as the sigmoid and hyperbolic tangent functions. Training a multilayer perceptron (MLP) is typically performed using the gradient descent algorithm. In this approach, the gradient of the objective function (usually the loss function) with respect to the weights is utilized to update the weights. This means that we are trying to adjust the weights in a way that minimizes the objective function, allowing the model to have good predictive capabilities.

2.2 Bee Algorithm Optimization Algorithm

The Artificial Bee Colony (ABC) Optimization algorithm is an optimization algorithm inspired by the foraging behavior of honey bees in nature, first introduced by Karaboga in 2005. This algorithm simulates the bees' collective search and exploitation of optimal food sources and has been successfully applied to various optimization problems such as function optimization, clustering, and routing. In the ABC algorithm, the bee colony is divided into three main types:

1. Employed Bees: These bees search for food sources around the current locations they are assigned to and are responsible for enhancing the quality of these sources.

2. Onlooker Bees: These bees stay in the hive and choose food sources based on a probability that depends on the quality

of food sources found by employed bees.

3. Scout Bees: If a food source has not improved beyond a certain threshold, it is abandoned, and scout bees search for new sources in unexplored areas.

The ABC algorithm consists of three main phases: employed bee phase, onlooker bee phase, and scout bee phase. Each phase includes formulas and parameters for updating and selecting the best solutions. Employed Bee Phase each employed bee identifies a new food source (solution) in the neighborhood of its current position. The new position of an employed bee is updated as follows:

$$v_{ij} = x_{ij} + \emptyset_{ij} \times (x_{ij} - x_{kj}) \quad (2)$$

Where $v_{ij}$is the new position of the employed bee i in dimension j, $x_{ij}$is the current position of bee iii in dimension j, x_{kj} is a randomly chosen position of another bee k in dimension j, and $\emptyset_{ij}$ is a random number in the range [−1,1], which adjusts the direction and magnitude of the movement. If the new food source has a higher fitness (quality) than the current one, the bee moves to the new position; otherwise, it retains its original position. Onlooker Bee Phase select food sources based on a selection probability, calculated by the following equation:

$$\boldsymbol{p_i} = \frac{\boldsymbol{f_i}}{\sum_{j=1}^{N} \boldsymbol{f_j}} \quad (3)$$

Where p_i is the probability of selecting food source i, f_i is the fitness of food source iii, and N is the total number of available food sources. This probabilistic mechanism favors food sources with higher fitness, increasing their chances of being selected by onlooker bees. Scout Bee Phase Sources that have not improved for a certain number of cycles are abandoned, and scout bees search randomly for new food sources using the following equation:

$$x_{ij} = x_{min} + rand(0,1) \times (x_{max} - x_{min}) \quad (4)$$

Where x_{min} and x_{max} are the minimum and maximum values in dimension j, and $rand(0,1)$ is a random number in the range [0,1]. Fitness Evaluation and Solution Selection of each food source is determined based on the objective function of the optimization problem. For minimization problems, the fitness function f(x) can be computed as follows:

$$f(x) = \begin{cases} 1/(1+F(x)) & if \;\; F(x) \geq 0 \\ 1 + |F(x)| & if \; F(x) < 0 \end{cases} \quad (5)$$

Where F(x) is the value of the objective function at position x. The algorithm aims to find the position with the smallest F(x) for minimization problems or the highest fitness for maximization problems.

3. Proposed Methodology

In this paper, aiming to enhance network intrusion detection accuracy and reduce false positives, we initiate by normalizing the data and eliminating less significant data based on statistical methods in the preprocessing phase, followed by the classification stage. The innovation in this study lies in the classification of data using a feedforward neural network, where the gradient descent algorithm is employed to optimize weights and biases. However, this algorithm faces the limitation of getting stuck in local optima, preventing the neural network from achieving its maximum classification accuracy. To address this issue, we replace the gradient descent algorithm with the bee colony optimization algorithm, which searches exhaustively for optimal weights and biases to minimize network error.

The preprocessing stage in intrusion detection is essential for preparing data, allowing the model to perform optimally. Figure 1 illustrate several steps involves in the proposed method. First, data cleaning is performed to remove noise, irrelevant information, and handle missing data. For missing values, the method of averaging the two nearest neighbors is employed, ensuring proportionate values relative to the other data. Next, feature scaling is applied, where a min-max scaling approach normalizes features within a defined range, typically between 0 and 1, ensuring all features contribute evenly in distance-based algorithms. Dimensionality reduction, a key step in preprocessing, simplifies the data space by reducing

features. Principal Component Analysis (PCA) is utilized for this, which involves centering the data, calculating the covariance matrix, and selecting components based on eigenvalues. This process results in transformed data represented in the space of principal components, aiding in the training and performance of machine learning models. The dataset is split into training and testing sets, typically in a 70-30 ratio, ensuring no overlap to evaluate the model's unbiased performance accurately.

In our multilayer perceptron (MLP) neural network, the input layer receives the problem's features, passing them directly to the hidden layer(s), which processes and extracts complex information. The output layer produces final classification results, often with an activation function like softmax for classification. In forward propagation, inputs are weighted and combined with biases at each layer, where they pass through an activation function to produce outputs. However, training MLPs using gradient descent can lead to local minima issues, which limits the model's capacity to reach the global optimum. To overcome this, the bee colony optimization algorithm is introduced, which operates in parallel, exploring the weight space thoroughly to identify global minima. This enhances the training and predictive capabilities of the MLP model. The bee algorithm's success relies on several parameters, including the number of bees representing neural weights and biases, the number of cycles for iterative optimization, the number of search regions, and the scout bees exploring diverse areas of the search space. Additionally, parameters such as the distance between search locations and adjustment rates allow fine-tuning for optimal performance. The objective function of the bee algorithm is designed to minimize the classification error of the neural network, ensuring that the population of solutions with better performance advances in the optimization process. The classification error, as defined by 1-Accuracy, serves as the evaluation metric, guiding the optimization toward improved accuracy and reduced error in intrusion detection.

3.1 Dataset

The NSL-KDD dataset is one of the well-known datasets for intrusion detection in networks and systems. It is a modified and enhanced version of the KDD Cup 1999 dataset, which, due to its issues, is not used as a baseline dataset for intrusion detection. NSL-KDD has improvements, including a reduction in the number of samples and correction of intrusion labels to assist better intrusion detection models.

3.2 Evaluation

Metrics to assess the performance of the proposed method in this article, metrics such as accuracy, precision, recall, and the F1 score have been used. The accuracy metric evaluates all correct intrusion detections and correct non-intrusion detections relative to all identifications, referred to as the overall accuracy of the system. Precision is a metric that evaluates correct intrusion detections relative to the total of correct intrusion identifications and false intrusion identifications. Recall is a metric that evaluates correct intrusion detections relative to the total number of actual intrusions in the dataset. Finally, the F1 score provides the harmonic mean of precision and recall. Mathematically, the F1 score is the weighted average of precision and recall. The best value for F1 is 1, and the worst value is 0.

In Table 1, the equations related to the evaluation metrics are presented. In these equations, TP represents true positive (correct intrusion detection), TN represents true negative (correct non-intrusion detection), and the metrics FP and FN represent false positive and false negative intrusion detections and non-intrusion detections, respectively. All these metrics are evaluable using the confusion matrix.

Table 1. Equations related to the evaluation metrics

Evaluation Metrics
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F_1 Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

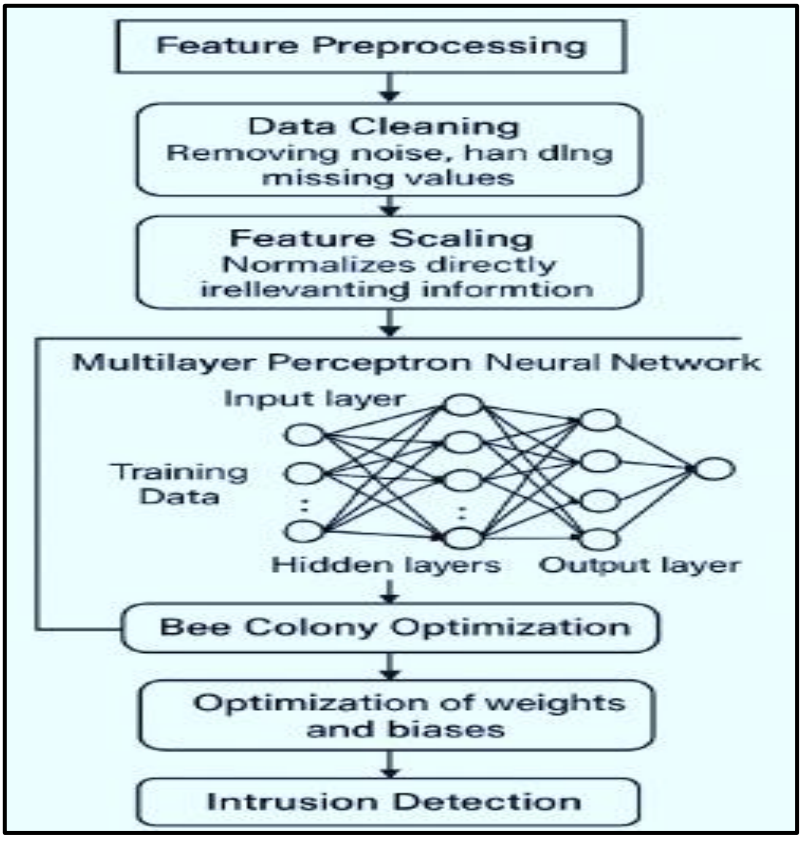

Figure 1. Digram of the proposed method

4. Results and Discussion

Simulation results: In the data preprocessing section, various actions were taken to transform the input data into a suitable and analyzable format for intrusion detection models. The actions performed by the MATLAB software in this section are explained. Initially, the data was loaded from the dataset, and then, using the ISNAN command available in MATLAB, missing data or values with no information were identified. Missing data might occur due to errors in measurement or other reasons. Subsequently, with the use of the KNNIMPUTE command and setting the value of K to 3, replacement values for missing data were calculated based on the average values of their 3 nearest neighbors and imputed. This process helped correct incomplete data and had a positive impact on data quality. In the next step, the data was normalized, meaning scaling the data to a specific range, typically between -1 and 1. This normalization ensures that data values are proportionate and regular, contributing to the improvement of intrusion detection models. The results of these stages are visualized in Figure 2, providing an insight into the impact of this preprocessing on the data and its distribution.

In the subsequent step, after performing the data preprocessing steps, the Principal Component Analysis (PCA) algorithm was employed for dimensionality reduction and selecting important features. The main goal of this stage was to select 33 out of the 41 features in the dataset with high importance. PCA operates by first calculating the covariance matrix of the data using mathematics and statistics. Then, based on this covariance matrix, the values of vectors and principal components (recognized as principal components) are computed. These principal components are associated with data features in a way that more important features are placed at the beginning of the list. The number of necessary components can then be selected based on their importance. In this case, 33 principal components with the highest importance were chosen, and 8

less important components were discarded. This action led to a reduction in data dimensions and the selection of important features, contributing to the enhancement of the quality and performance of the intrusion detection model.

	1	2	3	4	5	6	7
1	0	2.6198e-09	5.2396e-09	2.6198e-08	1.2863e-06	0	0
2	0	5.2396e-09	7.8594e-09	2.6198e-08	3.8249e-07	0	0
3	0	2.6198e-09	1.0479e-08	1.5719e-08	0	0	0
4	0	2.6198e-09	1.3099e-08	2.6198e-08	6.0779e-07	2.1359e-05	0
5	0	2.6198e-09	1.3099e-08	2.6198e-08	5.2134e-07	1.1003e-06	0
6	0	2.6198e-09	1.0479e-08	5.2396e-09	0	0	0
7	0	2.6198e-09	1.0479e-08	1.5719e-08	0	0	0
8	0	2.6198e-09	1.0479e-08	1.5719e-08	0	0	0
9	0	2.6198e-09	1.5719e-08	1.5719e-08	0	0	0
10	0	2.6198e-09	1.0479e-08	1.5719e-08	0	0	0
11	0	2.6198e-09	1.0479e-08	5.2396e-09	0	0	0
12	0	2.6198e-09	1.0479e-08	1.5719e-08	0	0	0
13	0	2.6198e-09	1.3099e-08	2.6198e-08	7.5188e-07	5.8972e-06	0
14	0	2.6198e-09	5.2396e-09	2.6198e-08	8.7501e-07	0	0
15	0	2.6198e-09	1.8339e-08	1.5719e-08	0	0	0

Figure 2. Output data of the preprocessing stage

After completing the above process, the next step involved creating the architecture of the Multilayer Perceptron (MLP) neural network as shown in the Figure3. In this architecture, the number of input layer neurons was set to 33, corresponding to the reduced features. The number of hidden layer neurons was determined to be 13 based on trial and error, and finally, the number of output layer neurons was set to 2, corresponding to the number of classes. The activation function for the hidden layer was chosen as the sigmoid, and the activation function used in the output layer was also sigmoid.

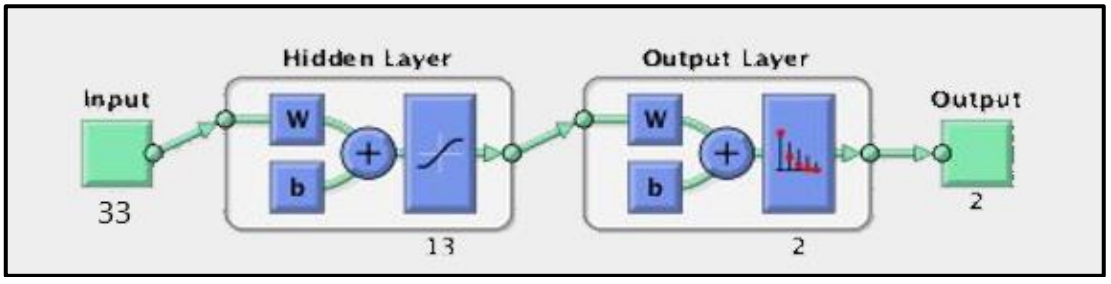

Figure 3. Architecture of the Multilayer Perceptron neural network used for classification

The Specifications of the Bee Algorithm is prsented in the Table 2, where its contain the parameters Initial Population, Number of Iterations, Lower Bound for Search, Upper Bound for Search, Upper Bound for Search and Termination Condition.

Table 2. Bee Algorithm Specifications

Parameter	Value
Initial Population	20
Number of Iterations	50
Lower Bound for Search	10
Upper Bound for Search	10
Objective Function	Increase Classification Accuracy or Reduce Error Rate
Termination Condition	Achieve 50 Iterations
Structure of Elements	Based on Weight and Bias

In Figure 4, the convergence chart of the training algorithm for the neural network is provided. This chart illustrates how the classification error changes for each iteration of the training process by adjusting the weights and biases of the neural network using the Bee Algorithm. Initially, the error may be relatively large for each iteration because the neural network has not yet learned enough information from the training dataset, and the weights and biases have been randomly set. Over time and with more iterations, it can be observed that the error gradually decreases. This indicates that the classifier, through the training process using the dataset, has fully learned and adjusted the weights and biases correctly. Finally, the error decreases to a level close to zero, indicating that the model has learned well and converged to an accurate and reliable classifier. This chart shows that the Bee Algorithm has helped the neural network reach a balanced and stable state in the training process, functioning as a strong classifier for intrusion detection in networks.

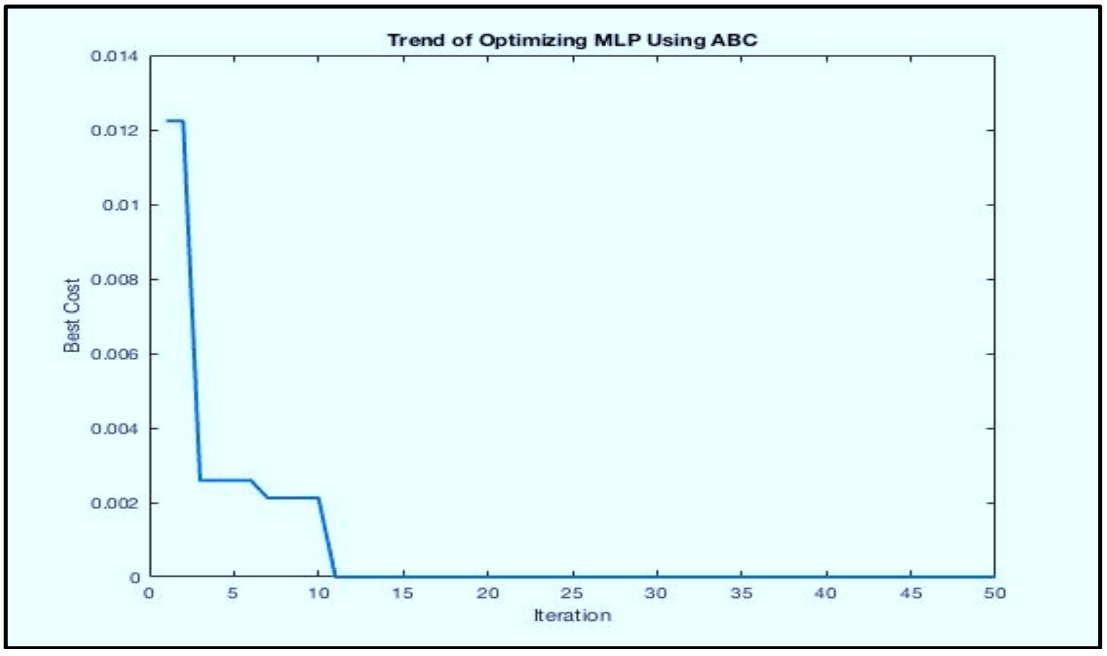

Figure 4. Convergence Curve of the Bee Algorithm

Examining the performance of the classifier model using the Confusion Matrix: Evaluating the performance of a classifier model using the Confusion Matrix is an essential method in intrusion detection and data classification problems. The Confusion Matrix serves as a powerful tool for assessing the accuracy and performance of a classification model. The following explanation details how the Confusion Matrix can be utilized to analyze the performance of the classifier model: The Confusion Matrix is a two-dimensional table consisting of four cells:

- True Positive (TP): The number of positive samples correctly identified.
- False Positive (FP): The number of negative samples mistakenly identified as positive.
- True Negative (TN): The number of negative samples correctly identified.
- False Negative (FN): The number of positive samples mistakenly identified as negative.

In Figure 5, the Confusion Matrix obtained for the test dataset of the proposed method is provided along with a table analyzing the results of this matrix.

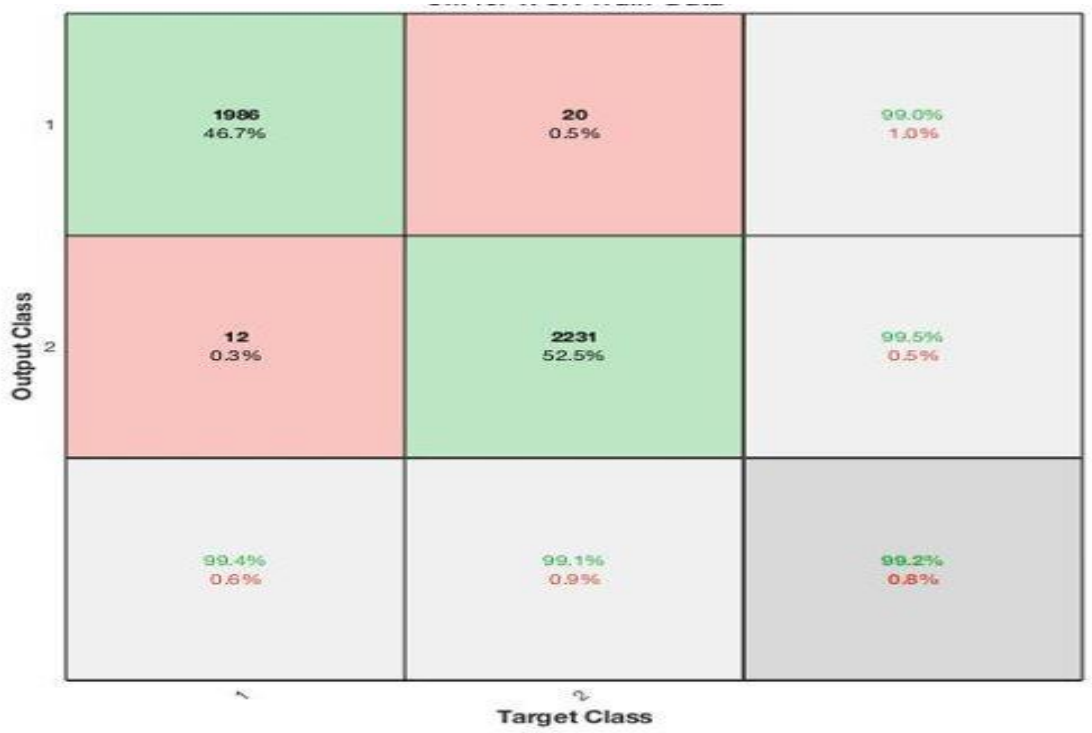

Figure 5. Confusion Matrix for the Test Dataset of the Perceptron Classifier

Table 3. Analysis of the Confusion Matrix for the Test Dataset

No.	Metric	Value
1	Accuracy	99%
2	False Intrusion Detection Rate	0.5%
3	True Intrusion Detection Rate	46.7%
4	True Negative Rate	99.5%
5	True Non-Intrusion Detection Rate	52.5%
6	False Non-Intrusion Detection Rate	0.3%
7	Precision	99.2%
8	Specificity	99.1%
9	Sensitivity and Recall	99.4%

Considering the confusion matrix provided in Figure 5 and the analysis, it is observed that the accuracy value obtained is also 99.2%.

- Examining the Performance of the Test Set Classifier Using the ROC Curve

This curve illustrates in detail how the model can distinguish between positive and negative classes and assesses the true and false detection at various thresholds. The horizontal axis in the ROC curve graph represents the false-positive rate, and the vertical axis represents the true-positive rate. In this curve, three regions are separated by the 45-degree line, and each of these regions indicates a specific scenario. For example, if the curves in the output graph that represents the classifier output are above the 45-degree line, it indicates a higher true-positive rate compared to false positives. If they are below the line, it indicates a higher false-positive rate. If they are on the line, it suggests an equal true-positive and false-positive rate. According to the information provided above, the region above the 45-degree line is the desirable region. Considering Figure 6, it can be concluded that the true-positive rate is higher than the false-positive rate in the proposed method, indicating the favorable performance of the proposed approach.

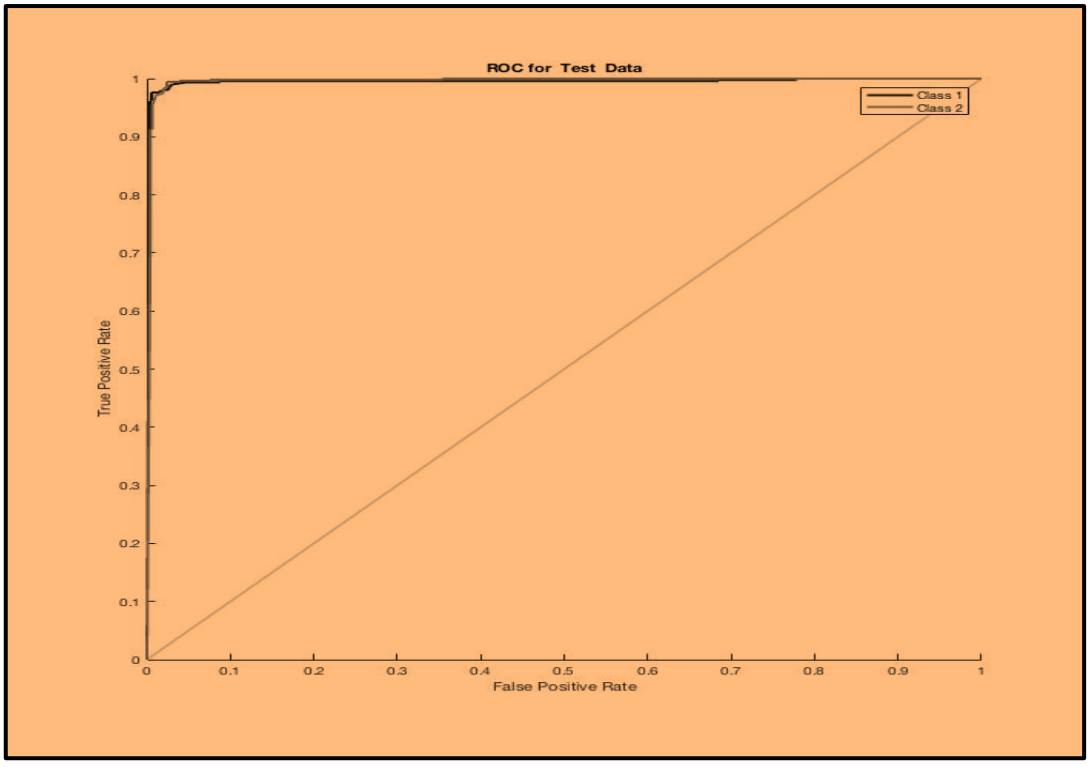

Figure 6. ROC Curve illustrating the performance of the test data

- Overall Test Dataset Results Analysis

Table 4 presents the evaluation metric results for the test dataset, allowing us to comprehensively assess the results from various perspectives.

Table 4. Evaluation Metrics Results for the Test Dataset

Metric	**Precision**	**Recall**	**Accuracy**
Proposed Method	99.2%	99.4%	99.1%

The result of the proposed method was compared with the state-of-the-art approaches. The comparison of the results obtained for the proposed model with other results of different methods is presented in Table 5. According to table 5, the proposed method demonstrates a 3% improvement in accuracy compared to the baseline approaches. The reason for this improvement lies in the utilization of the artificial bee colony algorithm to find optimal weights and biases instead of using the gradient descent algorithm. This algorithm, through its search process, can optimize the weights and biases effectively, leading to a significantly higher classification accuracy. The comparisons were performed on the same database across all methods. The proposed method could be applied into state of art systems that use microcontroler in diffrent application such as medical monetoring smart home monetoring and other application contain important informations[11-13].

Table 5. Comparison between the Proposed Method and Other Models

No.	Method	Accuracy	Ref, Year
1	SVM + GWO	96%	[10],2021
2	SVM + PSO	89%	[10], 2021
3	DNN	98.7%	[14], 2022
4	MLP + ABC	99.1%	This work

5. Conclusion

This research demonstrates the effectiveness of combining a multi-layer perceptron (MLP) neural network with the artificial bee colony (ABC) optimization algorithm for intrusion detection in computer networks. The MLP, with its ability to learn complex patterns and generalize new attack types, plays a crucial role in detecting intrusions. By optimizing the network's weights and biases using the ABC algorithm instead of traditional gradient descent, detection accuracy significantly improved. The proposed approach achieved an accuracy of 99.1%, which is approximately by an average 3% higher than state of art methods considered that used SVM+PSO, SVM+GWO and DNN models. This research highlights the potential of hybrid machine learning techniques in advancing network security and safeguarding sensitive data.

References

[1] Shun, J., & Malki, H. A. (2008). Network intrusion detection system using neural networks. In *Proceedings of the Fourth International Conference on Natural Computation* (Vol. 5, pp. 242–246). IEEE.
[2] Shenfield, A., Day, D., & Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks. *ICT Express, 4*(2), 95–99.
[3] Vinchurkar, D. P., & Reshamwala, A. (2012). A review of intrusion detection system using neural network and machine learning. *Journal of Engineering Science and Innovative Technology, 1*, 54–63.
[4] Debar, H., Becker, M., & Siboni, D. (1992). A neural network component for an intrusion detection system. In *Proceedings of the IEEE Symposium on Security and Privacy* (pp. 240–250).
[5] Mohammadpour, L., Ling, T. C., Liew, C. S., & Chong, C. Y. (2018). A convolutional neural network for network intrusion detection system. *Proceedings of the Asia-Pacific Advanced Network, 46*, 50–55.
[6] Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications, 36*(1), 16–24.
[7] Imrana, Y., Xiang, Y., Ali, L., Noor, A., Sarpong, K., & Abdullah, M. A. (2024). CNN-GRU-FF: A double-layer feature fusion-based network intrusion detection system using convolutional neural networks and gated recurrent units. *Complex & Intelligent Systems, 10*(3), 3353–3370.
[8] Gupta, C., Kumar, A., & Jain, N. K. (2024). An enhanced hybrid intrusion detection based on crow search analysis optimizations and artificial neural network. *Wireless Personal Communications, 134*(1), 43–68.
[9] Osa, E., Orukpe, P. E., & Iruansi, U. (2024). Design and implementation of a deep neural network approach for intrusion detection systems. *e-Prime: Advances in Electrical Engineering, Electronics and Energy, 7*, 100434.
[10] Vibhute, A. D., et al. (2024). An LSTM-based novel near-real-time multiclass network intrusion detection system for complex cloud environments. *Concurrency and Computation: Practice and Experience, 36*(11), e8024.
[11] Qasim, H., Hamza, A., Ibrahim, H., Saeed, H., & Hamzah, M. (2020). Design and implementation home security system and monitoring by using wireless sensor networks (WSN) / Internet of Things (IoT). *International Journal of Electrical and Computer Engineering (IJECE), 10*(3), 2617–2624.
[12] Hamza, A., et al. (2020). Design and implement WSN/IoT smart parking management system using microcontroller. *International Journal of Electrical and Computer Engineering, 10*(3), 3108.
[13] Saeed, H., et al. (2019). IoT health monitoring system for preventing and controlling risk in confined space using microcontrollers. *International Journal of Engineering and Technology, 8*(4), 619.
[14] Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T., & Bahaj, S. A. (2022). Deep learning for intrusion detection and security of Internet of Things (IoT): Current analysis, challenges, and possible solutions. *Security and Communication Networks, 2022*, 4016073.