Wasit Journal for Pure Science

Journal Homepage: https://wjps.uowasit.edu.iq/index.php/wjps/index
e-ISSN: 2790-5241 p-ISSN: 2790-5233

Intelligent Cyber-Attack Detection in IoT Networks Using IDAOA-Based Wrapper Feature Selection

Mohammed Abdullah Amer[1]*, Ryna Svyd[2]

[1]Wasit health Directorate, Kut, IRAQ
[2]Vasyl Stefanyk Precarpathian National University, The department of Computer Engineering and Electronics, Ukraine

*Corresponding Author: Mohammed Abdullah Amer

DOI: https://doi.org/10.31185/wjps.731
Received 12 January 2025; Accepted 18 March 2025; Available online 30 Jun 2025

ABSTRACT: In the realm of cybersecurity, the increasing sophistication of cyber-attacks demands the creation of sophisticated intrusion detection systems (IDS) designed to accurately detect and counteract threats in real-time. This study presents an innovative framework that integrates the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) with a Bagging technique to enhance the performance of intelligent cyber intrusion detection systems. The IDAOA serves as a wrapper-based feature selection method, optimizing the identification of the most impactful features while balancing local exploration and global exploitation. The Bagging technique further strengthens the classification phase by combining predictions from multiple classifiers, effectively addressing issues of class imbalance and improving overall system robustness. Evaluation of the proposed system using the NSL-KDD dataset demonstrates its superior performance, achieving an accuracy of 99.45%, significantly outperforming state-of-the-art approaches. Moreover, the proposed method achieved Precision, Recall, and F-score values of 99.63, 99.52, and 99.57, respectively, indicating its high reliability. These findings underscore the potential of intelligent optimization and ensemble learning techniques in advancing cybersecurity for IoT networks.

Keywords: Cybersecurity, Intrusion Detection System (IDS), Machine Learning, Optimization Algorithms

1. INTRODUCTION

The advent of the Internet of Things (IoT), combined with the advanced communication capabilities of 5G and 6G technologies, has revolutionized remote data monitoring and automated decision-making processes [1,2]. While these advancements have enhanced operational efficiency, they have also exposed IoT networks to cybersecurity threats, such as unauthorized access and data breaches. Network Intrusion Detection Systems (NIDS) have thus become crucial for combating threats like denial-of-service (DoS), distributed denial-of-service (DDoS), and sophisticated malware [3,4]. Traditional NIDS, which rely on signature-based or anomaly-based detection methods, are effective to some extent but struggle to address advanced threats such as zero-day exploits and polymorphic malware [5-7]. The rapid growth of IoT devices and cloud computing demands more adaptive and intelligent intrusion detection methods [8]. Machine learning (ML)-based NIDS have shown transformative potential by identifying emerging threats and dynamic patterns those conventional systems might overlook [9-11]. However, the success of ML-based NIDS heavily depends on the availability of high-quality training data, which remains a critical challenge [12-14]. The integration of IoT and cloud computing in domains such as healthcare and smart homes has highlighted the need for efficient anomaly detection systems. In [15], a lightweight and accurate anomaly detection model for IoT-cloud environments was proposed.

In [16], an explainable artificial intelligence (XAI)-enabled hybrid transfer learning model was introduced. This model leverages BiLAE and BMO algorithms to optimize feature extraction and hyperparameters, enabling effective zero-day attack detection even with limited labeled data. Similarly, [17] proposed a WSN IDS combining the CatBoost classifier and the Lyrebird Optimization Algorithm (LOA), achieving high detection accuracy for wireless sensor networks. Deep learning-based approaches have also been effective for anomaly detection in Industrial IoT (IIoT)

8*Corresponding author: author@organization.edu.co

networks. In [18], a deep transfer learning model was developed using the EdgeIIoT dataset to detect 14 distinct attack types. Resource-efficient techniques, such as model quantization, were employed to enhance performance. In Flying Ad-hoc Networks (FANETs), [19] addressed real-time detection of DDoS attacks. In [20], a novel intrusion detection model for connected autonomous vehicle (CAV) networks was proposed, combining deep learning and optimization techniques to handle high-dimensional data and noisy features. Additionally, [21] explored the use of GAN-generated synthetic data for training ML models in NIDS, reducing reliance on real-world datasets. Feature selection techniques have been employed to reduce computational complexity and enhance detection accuracy [22]. In [23], a novel approach transformed network traffic into audio signals, effectively capturing intricate patterns and anomalies. Similarly, [24] introduced an IDS combining particle swarm optimization (PSO) with the AdaBoost algorithm to identify key features and improve detection accuracy. Finally, [25] proposed a two-phase IDS that integrates supervised and unsupervised learning methods to detect a wide range of cyberattacks with high accuracy and low false alarm rates.

This study addresses these challenges by introducing an Intelligent Cyber-Attack Detection System (ICADS) that leverages advanced optimization and ensemble learning techniques. The proposed system integrates the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) for feature selection and the Bagging ensemble method for classification. IDAOA enhances the efficiency and accuracy of feature selection through adaptive mechanisms, while the Bagging method improves resilience to noise and imbalanced data. Together, these advancements provide a robust framework capable of detecting both known and emerging threats in IoT networks. By bridging the gap between dynamic threat landscapes and static security measures, the proposed ICADS represents a significant step forward in securing modern network infrastructures. The key contributions of this research are summarized as follows:

- Introducing an efficient wrapper-based metaheuristic optimization framework that leverages enhanced exploration capabilities to identify and select the optimal subset of features, maximizing their impact on attack detection in intelligent cyber intrusion detection systems.
- Proposing a new optimization algorithm called the Improved Dynamic Arithmetic Algorithm (IDAOA) with dynamic inertia weights and adaptive mutation coefficient to reduce the risk of convergence to local optima in selecting the optimal feature subset in intelligent intrusion detection systems.

This paper investigates the challenges associated with detecting cyber-attacks in IoT networks and proposes an Intelligent Cyber-Attack Detection System (ICADS). The remainder of this paper is structured as follows: In Section 2, the proposed methodology is described, with a focus on the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) and the Bagging technique for feature selection and classification. In Section 3, the evaluation results are presented, demonstrating the effectiveness of the proposed system. Finally, in Section 4, the study is concluded, and potential directions for future research are discussed.

2. PROPOSED METHOD

The objective of this study is to present an effective and efficient model for intrusion detection by integrating the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) with the Bagging technique for classification. In the feature selection phase, IDAOA is employed as a wrapper method. It leverages a balanced search capability between local exploration and global exploitation to identify the optimal subset of features. These selected features play a pivotal role in reducing the model's complexity and enhancing the accuracy of the intrusion detection system. The proposed IDAOA incorporates dynamic inertia weights to accelerate convergence speed. Additionally, the use of dynamic coefficient of mutation and triangular mutation strategy enhances the algorithm’s ability to avoid local optima. These dynamic update operators for coefficients and mutation improve the algorithm’s adaptability to complex and dynamic datasets. For the classification phase, the Bagging technique is utilized to develop a robust and noise-resilient system, capable of handling imbalanced data effectively. By combining predictions from multiple classifiers, Bagging reduces generalization error and significantly enhances the system's accuracy and stability. This integration equips the model not only to excel in detecting known attack types but also to effectively identify new and unknown attack patterns. Figure 1 illustrates the proposed intrusion detection method using the IDAOA approach.

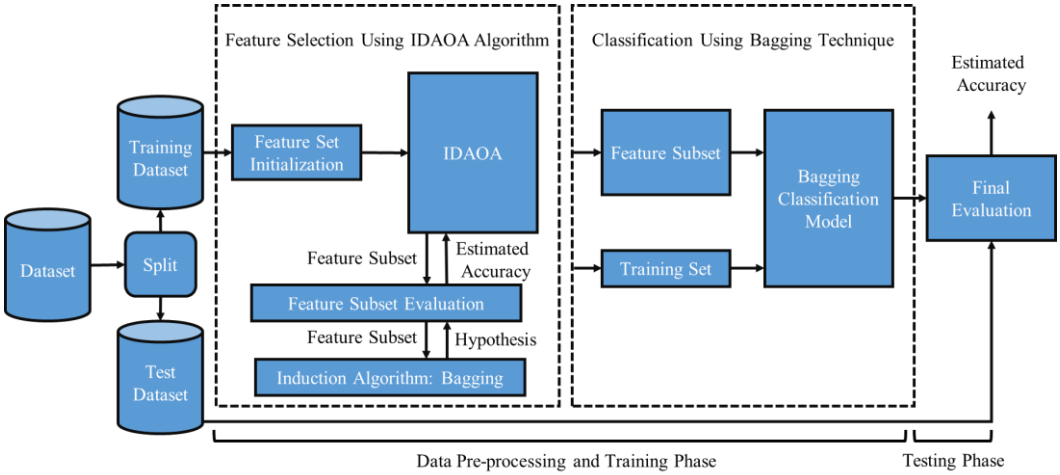

FIGURE 1. Presented Intrusion Detection Method using IDAOA based Wrapper feature selection

2.1 DATA PREPROCESSING

In the preprocessing stage of the proposed method, data normalization is performed using the Min-Max technique. This method aims to normalize feature values within the range [0,1], reducing the influence of differing feature scales on the effectiveness of learning algorithms. Min-Max normalization ensures faster and more precise optimization while improving the classification model's accuracy and stability. The Min-Max normalization is computed using the following equation:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Where, x is the original value of the data, x_{min} denotes the minimum value of the feature across all data, x_{max} is the maximum value of the feature across all data, and x_{norm} represents the normalized feature value. The transformation described maps original data values to the range [0,1], scaling the smallest value to 0 and the largest to 1, with intermediate values adjusted linearly. This Min-Max normalization ensures all features contribute equally during training, preventing larger scales from dominating. A key benefit is that it preserves the original data distribution, which is crucial for algorithms like Bagging, as it enhances model performance. By using this method, the model analyzes normalized data more accurately, reduces noise from feature scale differences, and improves intrusion detection precision.

2.2 FEATURE SELECTION USING IDAOA

The aim of this work is to identify an optimal subset of features using the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) to maximize the accuracy of the proposed method. IDAOA is an optimization algorithm based on arithmetic operators such as addition, subtraction, multiplication, and division. This algorithm operates through two primary phases: exploration and exploitation. Figure 2 presents the IDAOA flowchart, and the following outlines the algorithm steps for identifying the optimal feature subset.

Step 1: Initialization

In this step, the optimization variables of IDAOA, namely α and μ, are initialized. The parameter α, which is a control parameter that determines the exploitation precision at each iteration, is set to 5 Similarly, the parameter μ, a control variable regulating the search process, is initialized to 0.5.

Step 2: Initial Potential Solutions

In the IDAOA algorithm, every initial solution X signifies a possible solution to addressing the problem, particularly the selection of optimal features for intrusion detection. Each initial solution X contains indices of the features considered optimal and is iteratively updated by the algorithm to approach the optimal solution. The initial population of solutions in the IDAOA algorithm is generated randomly within the problem space and updated in each iteration. Candidate initial solutions are produced randomly according to Equation (2). Subsequently, the best candidate solution at each iteration is identified and regarded as the best or near-optimal solution so far. In this work, the feature indices are treated as X.

$$X = \begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,j} & x_{1,n-1} & x_{1,n-1} \\ x_{2,1} & \cdots & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & x_{N-1,j} & \cdots & x_{N-1,n} \\ x_{N,1} & \cdots & \cdots & x_{N,j} & x_{N,n-1} & x_{N,n} \end{bmatrix} \quad (2)$$

Here, each candidate solution is gradually refined toward the optimal solution through successive iterations.

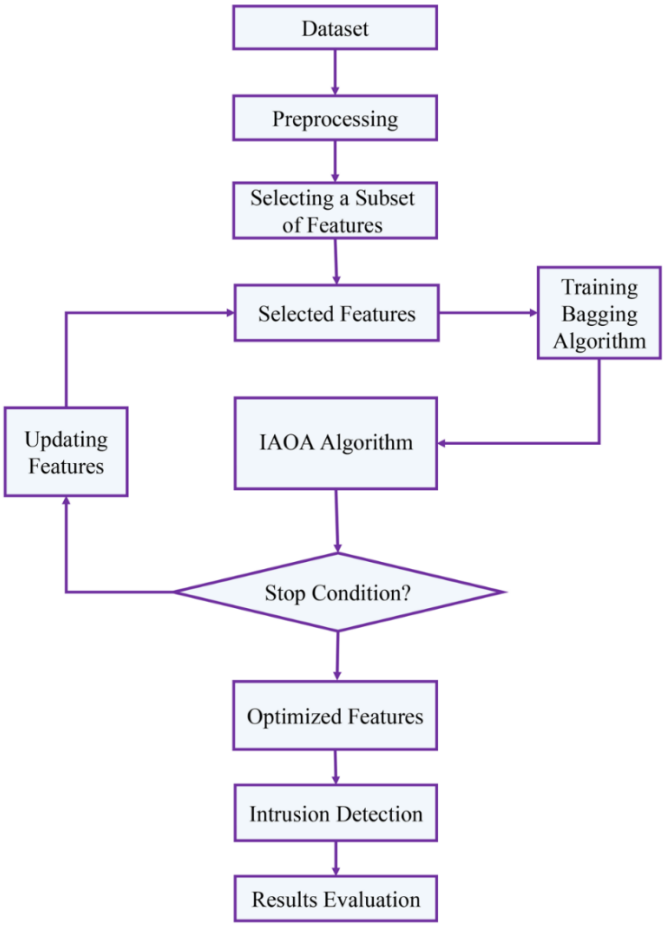

FIGURE 2. Feature Selection

Step 3: Calculating the Fitness Function

A critical aspect of optimization algorithms lies in defining the objective function. Here, we describe the process of formulating the objective function for IDAOA to identify an optimal feature subset. The fitness function employed in this work is defined as the accuracy of intrusion detection achieved using the Bagging algorithm. Specifically, a subset of features identified by the IDAOA algorithm is fed into the Bagging algorithm. The Bagging algorithm then performs intrusion detection using these features. The detection accuracy is computed and used as the fitness value, expressed as:

$$fitness = Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

In the equation above, TP, TN, FP, and FN represent True Positive, True Negative, False Positive, and False Negative, respectively.

Step 4: Identifying the Best Solution

Based on the fitness values calculated in the previous step, the best solution identified so far is determined in this step. This involves selecting the solution with the highest fitness value up to the current iteration, ensuring optimal feature subset selection.

Step 5: Updating the Math Optimizer Acceleration (MOA)

Before starting the AOA, the search phase (either exploration or exploitation) must be determined. The acceleration function of the Math Optimizer is calculated using the following equation:

$$MOA(C\_Iter) = Min + C\_Iter \left(\frac{Max-Min}{M\_Iter}\right) \quad (4)$$

Where $MOA(C\_Iter)$ represents the value of the acceleration function at iteration t, calculated using the above equation. C_Iter represents the current iteration, which varies from 1 to the maximum iteration limit (M_Iter). Min and Max are the minimum and maximum values of the acceleration function, respectively.

Step 6: Updating the Mathematical Optimizer Probability (MOP)

The MOP probability is updated using Equation (5):

$$M\ O\ P(C_{Iter}+1) = 1 - \frac{C_{Iter}^{\frac{1}{\propto}}}{M_{Iter}^{\frac{1}{\propto}}} \quad (5)$$

Where, $M\ O\ P(C_{Iter})$is the value of the MOP probability function at the current iteration. M_{Iter} is the maximum count of iterations. (C_{Iter}) denotes the current iteration.

Step 7: Solution Position Updates

In this step, the following procedure is repeated for all solutions (i.e., all members of population X). First, three random numbers between 0 and 1, denoted as r_1, r_2 and r_3, are generated. If, for any given solution, r_1 is greater than the MOA value, the exploration phase is executed; otherwise, the exploitation phase is carried out.

Step 8: Exploration Phase

If $r_2 < 0.5$, the division operator (D) is applied. In this case, the solution position is updated according to the first rule in the following equation. Otherwise, the multiplication operator (M) is applied, and the solution position is updated according to the second rule:

$$x_{i.j}(C_{Iter}+1) = \begin{cases} best\ (x_i) \div (M\ O\ P + \in) \times \left(\left(UB_j\right) - LB_i\right) \times \mu \times LB_i), & r_2 < 0.5 \\ best\ \left(x_j\right) \times M\ O\ P \times \left(\left(UB_j\right) - LB_i\right) \times \mu \times LB_i), & otherwise \end{cases} \quad (6)$$

Where: $x_i(C_{Iter}+1)$ is the position of the j-th dimension of solution i at the next iteration. $x_{i.j}(C_{Iter})$ denotes the position of the j-th dimension of solution i in the current iteration. $best\ (x_i)$ represents the best solution found so far. $\in$ is a constant. UB_j and LB_j are the upper and lower bounds for the j-th dimension.

Step 9: Exploitation Phase

If $r_3 < 0.5$, the subtraction operator (S) is applied, updating the solution position based on the first rule in Equation (7). Otherwise, the addition operator (A) is applied, and the position is updated based on the second rule:

$$x_{i.j}(C_{Iter}+1) = \begin{cases} best\ \left(x_j\right) - M\ O\ P \times \left(\left(U\ B_j\right) - LB_j\right) \times \mu + LB_j), & r3 < 0.5 \\ best\ \left(x_j\right) + M\ O\ P \times \left(\left(U\ B_j\right) - LB_j\right) \times \mu \times LB_j), & otherwise \end{cases} \quad (7)$$

Step 10: Dynamic Inertia Weights Update

The AOA algorithm is susceptible to getting stuck in local optima and exhibits slow convergence during the search, as it updates solutions solely based on the global best position. To address this, the IDAOA algorithm employs dynamic inertia weights to improve the convergence speed of AOA. Dynamic inertia weights directly control the rate of solution updates during each optimization step. Larger inertia weights at the initial stages of the algorithm reduce rapid changes in potential solutions X within the search space, allowing for a broader exploration. Conversely, smaller inertia weights in the later stages limit the movement of solutions within a narrower range, enhancing exploitation. In this work, nonlinear, exponentially decreasing dynamic inertia weights are introduced to enhance the AOA algorithm's search efficiency, thereby improving convergence speed. The dynamic inertia weights are calculated as follows:

$$w(t) = c \times w_{begin} \left(\frac{w_{begin}}{w_{end}}\right)^{\frac{1}{\left(\frac{1+t}{T}\right)}} \quad (8)$$

Where:

w_{begin} and w_{end} represent maximum and minimum inertia weights, respectively. C is a random constant dynamically varying around 1.t is current iteration. T denotes Maximum number of iterations. By integrating dynamic inertia weights, the position update equations (6) and (7) in the AOA algorithm are modified as follows:

$$x_{i.j}(C_{Iter}+1) = \begin{cases} w(t) \times best\ (x_i) \div (MOP + \in) \times \left(\left(UB_j\right) - LB_i\right) \times \mu \times LB_i), & r_2 < 0.5 \\ w(t) \times best\ \left(x_j\right) \times MOP \times \left(\left(UB_j\right) - LB_i\right) \times \mu \times LB_i), & otherwise \end{cases} \quad (9)$$

$$x_{i.j}(C_{Iter}+1) = \begin{cases} w(t) \times best\ \left(x_j\right) - MOP \times \left(\left(UB_j\right) - LB_j\right) \times \mu + LB_j), & r3 < 0.5 \\ w(t) \times best\ \left(x_j\right) + MOP \times \left(\left(UB_j\right) - LB_j\right) \times \mu \times LB_j), & otherwise \end{cases} \quad (10)$$

Step 11: Dynamic Mutation

This study presents a dynamic mutation coefficient that grows as the number of iterations increases, providing individuals with a specific probability to explore different search spaces. Employing this approach effectively broadens the search domain and enhances the algorithm's ability to escape local optima. The calculation method for the dynamic coefficient of mutation is provided in Equation (11).

$$p = 0.2 + 0.5 \times \frac{t}{T} \quad (11)$$

Where: p represents Dynamic mutation probability, which grows as the number of iterations increases.

In this study A triangular mutation strategy is applied to diversify the population. This strategy prevents the algorithm from getting trapped in a local optimum during the search process. This strategy involves selecting three random solutions and combining them as follows:

$$X(t) = \frac{X_{r1}+X_{r2}+X_{r3}}{3} + (p2 - p1) \times (X_{r1} - X_{r2}) + (p3 - p2) \times (X_{r2} - X_{r3}) + (p1 - p3) \times (X_{r3} - X_{r1}) \quad (12)$$

Here: X_{r1} , X_{r2} and X_{r3} denote Randomly selected solutions. $(p2 - p1)$, $(p3 - p2)$ and $(p1 - p3)$ represent weights-perturbed patterns, $p1$, $p2$ and $p3$ are calculated as:

$$p1 = \frac{|f(X_{r1})|}{\acute{p}} \quad (13)$$

$$p2 = \frac{|f(X_{r2})|}{\acute{p}} \quad (14)$$

$$p3 = \frac{|f(X_{r3})|}{\acute{p}} \quad (15)$$

The triangular mutation strategy ensures the integration of information from randomly selected solutions, increasing population diversity and reducing the likelihood of getting trapped in local optima.

Step 12: Termination Condition

Steps 3 to 11 are repeated until a termination condition is met. In this work, the termination condition is defined as reaching the maximum number of iterations.

2.3 CLASSIFICATION USING BAGGING ALGORITHM

This research utilizes the Bagging (Bootstrap Aggregating) technique to improve the reliability and accuracy of intrusion detection. Bagging, a prominent ensemble learning technique, is developed to reduce model variance and improve generalizability. It achieves this by randomly sampling subsets of the original data with replacement and training separate classifiers on these subsets. Each classifier thus observes only a portion of the dataset and constructs its model accordingly [26]. In the proposed method, subset selection is performed through sampling with replacement, meaning that individual data points may appear in multiple subsets. This strategy ensures diversity among classifiers, reducing the risk of high correlation between models. By mitigating the effects of random fluctuations in the data and resisting overfitting, Bagging provides superior accuracy in identifying various intrusion attack types. Furthermore, Bagging consolidates predictions from multiple classifiers to form a stronger and more stable final model, thereby improving the intrusion detection system's performance when handling complex and diverse datasets. These features make Bagging an ideal choice for cybersecurity applications in dynamic and heterogeneous environments. Figure 3 provides an illustration of the overall functionality of this framework.

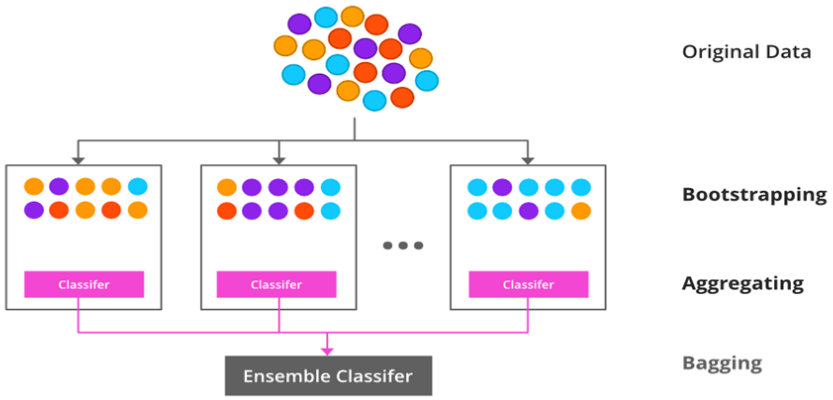

FIGURE 3. Classification Using Bagging Technique

2.4 Computational Complexity of the IDAOA Optimization Algorithm

The computational complexity of the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) is a function of the number of particles (population size) N, the problem dimensions (number of variables) d, and the number of iterations I, and is expressed as follows:

$$Complexity\ (IDAOA) = O(f(N.d.I)) \quad (16)$$

Thus, the computational complexity of this algorithm is analyzed in the following five stages:

- Initialization

In this stage, the initial population of NNN solutions is randomly generated. This process has a complexity of O(N·d) since each of the N particles must be assigned an initial value in a d-dimensional space.

- Fitness Evaluation

Each member of the population must be evaluated using the objective function. Assuming that the objective function has a computational complexity of O(F), the complexity of this stage is given by O(N·F).

- Particle Position Update

During each iteration, the IDAOA algorithm updates the position of particles using arithmetic operations (addition, subtraction, multiplication, and division). This process is performed for all N particles across d dimensions, resulting in O(N·d) complexity.

- Termination Condition Check and Iterations Execution

The algorithm runs for a maximum of III iterations. Therefore, the overall complexity at this stage is O(I·(N·d+N·F)).

- Final Complexity

The computational cost of the fitness function, denoted as FFF, depends on the number of features or problem dimensions. In the worst-case scenario, O(F) is equivalent to O(d). Thus, the final complexity of the algorithm can be expressed as:

$$Complexity\ (IDAOA) = O(I \cdot N \cdot d) \qquad (17)$$

The efficiency of the IDAOA algorithm, in terms of computational complexity, depends on the population size, the number of dimensions, and the number of iterations. This complexity is comparable to other metaheuristic optimization algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). However, depending on the problem structure and initialization strategy, IDAOA may exhibit superior convergence performance.

3. SIMULATIONS AND RESULTS

This section evaluates the proposed method's efficiency leveraging parameters such as Recall, F-Score, Precision and Accuracy. The efficacy of the proposed method is evaluated in comparison to other leading techniques. The simulations were conducted using MATLAB (2022b) on a system equipped with an NVIDIA GPU, 32 GB RAM, an Intel Core i7 processor, and a Windows operating system. Additionally, classification accuracy was measured using 10-fold cross-validation, where the model was trained and tested iteratively with each fold serving as the validation set. The results presented in the tables represent the averages of 50 independent runs of the program.

3.1 DATASET

This work utilizes the NSL-KDD dataset, an improved version of the KDD'99 dataset designed to address its predecessor's limitations. The NSL-KDD database comprises a total of 148,517 records, divided into 125,973 training samples and 22,544 testing samples. It features 41 attributes and categorizes network traffic into five classes: four attack classes (DoS, Probe, R2L, U2R) and one normal class. Table 1 summarizes the number of samples in each class for the training and testing datasets.

Table 1. Number of Samples in Each Class in KDD-Train and KDD-Test Datasets

Attack Types	Number of samples in KDD-train dataset	Number of Samples in KDD-test dataset
Normal	67343	9711
DoS	45927	7456
Probe	11656	2421
R2L	995	2756
U2R	52	200
Total	125973	22544

3.2 EVALUATION METRICS

In classification problems, classification accuracy is the primary criterion. The accuracy for a specific class is calculated as the proportion of correctly classified samples for that class relative to the total number of misclassified instances. The performance of the proposed method in this study is assessed using the metrics of Precision, Recall, F-score, and Accuracy. The mathematical formulas for calculating these metrics are presented in the following equations:

$$Accuracy(acc) = \frac{TP+TN}{TP+TN+FP+FN} \qquad (18)$$

$$precision = \frac{TP}{TP+FP} \qquad (19)$$

$$recall = \frac{TP}{TP+FP} \quad (20)$$

$$F1score = 2 * \frac{recall*precision}{recall+precision} \quad (21)$$

In the equations above, TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives.

3.3 EVALUATION RESULTS

Figure 4 illustrates the confusion matrix from simulating the proposed method, demonstrating its effectiveness in detecting intrusions across five classes: DoS, Normal, Prob, R2L, and U2R. The matrix shows actual classes in rows and predicted classes in columns. For the DoS class, out of 7,546 samples, 7,449 were correctly classified, resulting in an accuracy of 99.9%. The Normal class achieved 100% accuracy with all 9,711 samples correctly classified. In the Prob class, 2,332 out of 2,421 samples were accurately predicted, yielding 96.3% accuracy. The R2L class had 2,731 correct identifications out of 2,756 samples, resulting in 99.1% accuracy. Lastly, the U2R class saw 1,997 out of 2,000 samples correctly classified, demonstrating an accuracy of 98.5%. Overall, the method achieves high accuracy in identifying all classes, with four classes exceeding 98.7% accuracy and perfect accuracy for the Normal class, showcasing its capability in effective intrusion detection.

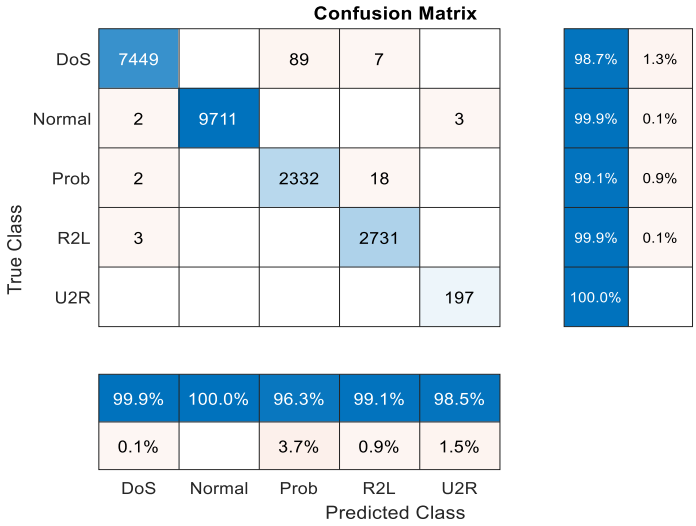

FIGURE 4. Confusion Matrix for Test Data

The Receiver Operating Characteristic (ROC) curve serves as a graphical representation to evaluate a classifier's effectiveness across different thresholds in machine learning. It is constructed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR), two critical performance metrics. TPR, also known as recall, is the proportion of all positive cases correctly identified. FPR represents the proportion of negative samples incorrectly classified as positive and is defined as follows:

$$FPR = \frac{FP}{TN+FP} \quad (22)$$

The trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) across different thresholds can be visualized by plotting them on an ROC curve. An effective classifier will produce a curve that gravitates toward the top-left corner, representing a high TPR and a low FPR. In contrast, a suboptimal classifier's curve tends toward the bottom-right corner, characterized by low TPR and high FPR. A random classifier generates a diagonal line on the ROC graph, indicating equal TPR and FPR values. As depicted in Figure 5, the ROC curve for our model closely approaches the top-left corner, signifying a high TPR and a low FPR. This observation underscores the model's exceptional ability to accurately detect intrusions.

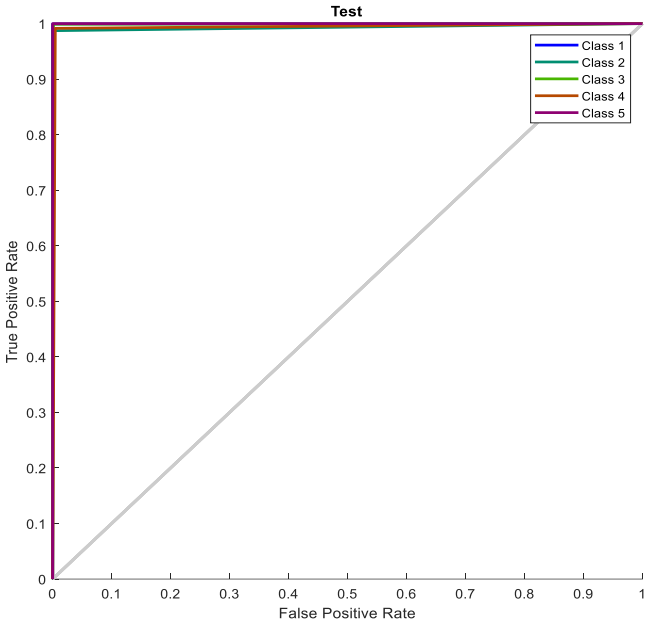

FIGURE 5. ROC Curve for the Proposed Method

Figure 6 illustrates a comparison of intrusion detection outcomes achieved by the presented method and several alternative approaches, including GAN-KNN, RF, XG-Boost, CNN, CNN-LSTM, KNN, Adaboost, and Naïve Bayes, using a bar chart that evaluates the performance based on Precision, Recall, and F-score. As shown in Figure 7, the proposed method outperforms all the other methods in terms of these metrics. The weakest performance in terms of Precision and F-score is observed in the GAN-KNN method. Furthermore, the weakest performance in terms of Recall is associated with the CNN-LSTM method. These results underline the superior performance of the proposed method across all evaluated criteria.

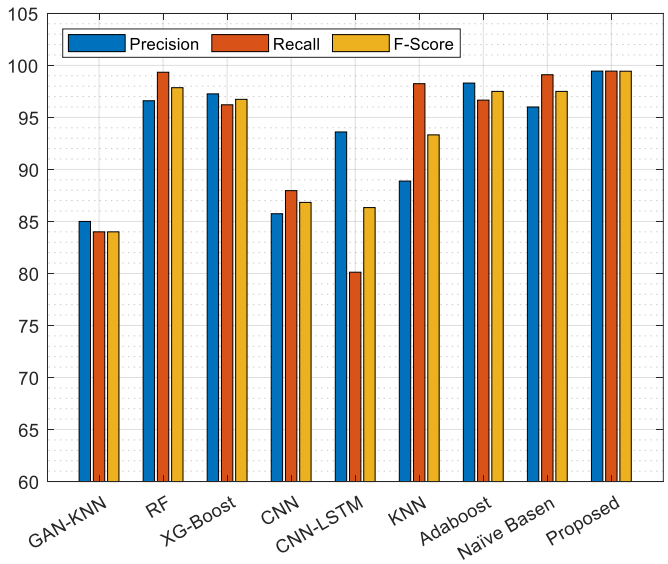

FIGURE 6. Comparison of the Proposed Method in Terms of Precision, Recall, and F-score with Other Methods

Table 2 presents a detailed comparison of the proposed method against alternative approaches for intrusion detection, emphasizing the Accuracy metric. As observed, the proposed method achieves the best performance with an accuracy of 99.45%, followed by the XG-Boost algorithm, which achieves 99.34% accuracy. The comparison highlights a minimum 1% improvement in the accuracy of the proposed method over all other methods, demonstrating its superior capability in accurately identifying intrusions.

Table 2. Results Comparison in Terms of Accuracy

Author	Method	Acc.
Rahman et al. [21]	GAN-KNN	84.00
Kumar et al. [22]	RF	98.33
Kumar et al. [22]	XG-Boost	99.34
Aldarwbi et al. [23]	CNN	84.82
Aldarwbi et al. [23]	CNN-LSTM	88.59
Sun et al. [24]	KNN	92.00
Sun et al. [24]	Adaboost	98.50
Vishwakarma et al. [25]	Naïve Base	97.10
Presented Method	Wrapper Based AOA + Bagging Classifier	99.45

In Table 3, the proposed method is compared with other optimization techniques in terms of accuracy criteria. As can be seen, the proposed method has the best performance with an accuracy of 99.45 percent. In addition, after the proposed method, the AOA algorithm has the best performance with an accuracy of 98.31 compared to other optimization algorithms. Also, the WOA, PSO, MPA and GA algorithms have achieved accuracies of 98.23, 97.94, 97.47 and 96.99 percent. Comparison of the results shows the superiority of the proposed algorithm over other optimization techniques.

Table 3. Results Comparison of proposed method against other optimization techniques in Terms of Accuracy

Method	Acc.
MPA	97.47
WOA	98.23
GA	96.99
PSO	97.94
AOA	98.31
Wrapper Based AOA	99.45

4. CONCLUSION

This study introduces a novel Intelligent Cyber-Attack Detection System (ICADS) designed to address critical challenges in safeguarding IoT networks from increasingly sophisticated cyber threats. The proposed system integrates the Improved Dynamic Arithmetic Optimization Algorithm (IDAOA) for feature selection and the Bagging ensemble method for classification, offering a comprehensive framework that enhances both efficiency and accuracy in intrusion detection. The IDAOA algorithm demonstrated exceptional capability in optimizing feature selection by leveraging dynamic inertia weights and a triangular mutation strategy. These mechanisms ensured a robust balance between exploration and exploitation, enabling the selection of a feature subset that significantly reduces computational complexity without compromising detection performance. Meanwhile, the Bagging method provided resilience against data imbalance and noise, further improving the system's generalization ability and robustness. The experimental results, validated on the NSL-KDD dataset, highlight the effectiveness of the proposed system, achieving an impressive accuracy of 99.45%. This performance surpasses several state-of-the-art methods, showcasing the system's superiority in handling both known and novel attack patterns. The findings underscore the potential of combining intelligent optimization techniques with ensemble learning to overcome inherent limitations in traditional intrusion detection systems. Future work will focus on expanding the system's applicability by testing it on real-time IoT environments and diverse datasets to evaluate its scalability and adaptability. Additional research could explore integrating advanced deep learning techniques with the ICADS framework to further enhance detection capabilities and address evolving cyber threats. This study contributes significantly to the field of cybersecurity, offering a practical and effective approach to strengthening the resilience of IoT networks against malicious activities.

REFERENCES

[1] M. Nassereddine and A. Khang, "Applications of Internet of Things (IoT) in smart cities," in *Advanced IoT Technologies and Applications in the Industry 4.0 Digital Economy*, CRC Press, 2024, pp. 109-136.

[2] M. Hossain, G. Kayas, R. Hasan, A. Skjellum, S. Noor, and S. R. Islam, "A Holistic Analysis of Internet of Things (IoT) Security: Principles, Practices, and New Perspectives," *Future Internet*, vol. 16, no. 2, p. 40, 2024.

[3] G. Gkagkas, D. J. Vergados, A. Michalas, and M. Dossis, "The Advantage of the 5G Network for Enhancing the Internet of Things and the Evolution of the 6G Network," *Sensors*, vol. 24, no. 8, p. 2455, 2024.

[4] O. H. Abdulganiyu, T. A. Tchakoucht, and Y. K. Saheed, "Towards an efficient model for network intrusion detection system (IDS): systematic literature review," *Wireless Networks*, vol. 30, no. 1, pp. 453-482, 2024.

[5] M. Almehdhar et al., "Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks," *IEEE Open Journal of Vehicular Technology*, 2024.

[6] R. Kimanzi, P. Kimanga, D. Cherori, and P. K. Gikunda, "Deep Learning Algorithms Used in Intrusion Detection Systems—A Review," *arXiv preprint arXiv:2402.17020*, 2024.

[7] P. Sanju, "Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks," *Journal of Engineering Research*, vol. 11, no. 4, pp. 356-361, 2023.

[8] Z. Azam, M. M. Islam, and M. N. Huda, "Comparative analysis of intrusion detection systems and machine learning based model analysis through decision tree," *IEEE Access*, 2023.

[9] S. V. N. Santhosh Kumar, M. Selvi, and A. Kannan, "A Comprehensive Survey on Machine Learning-Based Intrusion Detection Systems for Secure Communication in Internet of Things," *Computational Intelligence and Neuroscience*, vol. 2023, p. 8981988, 2023.

[10] M. A. Talukder et al., "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, no. 1, p. 33, 2024.

[11] V. Kukartsev et al., "Using machine learning techniques to simulate network intrusion detection," in *2024 International Conference on Intelligent Systems for Cybersecurity (ISCS)*, 2024, pp. 1-4.

[12] I. Hidayat, M. Z. Ali, and A. Arshad, "Machine learning-based intrusion detection system: an experimental comparison," *Journal of Computational and Cognitive Engineering*, vol. 2, no. 2, pp. 88-97, 2023.

[13] A. Awajan, "A novel deep learning-based intrusion detection system for IoT networks," *Computers*, vol. 12, no. 2, p. 34, 2023.

[14] R. Saadouni et al., "Intrusion detection systems for IoT based on bio-inspired and machine learning techniques: a systematic review of the literature," *Cluster Computing*, pp. 1-27, 2024.

[15] A. H. Farooqi, R. Ahmad, and S. Kamal, "ML-Driven Lightweight Botnet Detection System for IoT-Networks," unpublished.

[16] Y. K. Saheed and J. E. Chukwuere, "XAIEnsembleTL-IoV: A new eXplainable Artificial Intelligence ensemble transfer learning for zero-day botnet attack detection in the Internet of Vehicles," *Results in Engineering*, vol. 24, p. 103171, 2024.

[17] S. S. Abinayaa et al., "Securing the Edge: CatBoost Classifier Optimized by the Lyrebird Algorithm to Detect Denial of Service Attacks in Internet of Things-Based Wireless Sensor Networks," *Future Internet*, vol. 16, no. 10, p. 381, 2024.

[18] B. Ahmad, Z. Wu, Y. Huang, and S. U. Rehman, "Enhancing the security in IoT and IIoT networks: An intrusion detection scheme leveraging deep transfer learning," *Knowledge-Based Systems*, vol. 305, p. 112614, 2024.

[19] S. P. Priyadharshini and P. Balamurugan, "An Efficient DDoS Attack Detection and Prevention Model using fusion Heuristic Enhancement of Deep Learning Approach in FANET Sector," *Applied Soft Computing*, p. 112438, 2024.

[20] F. S. Alrayes et al., "Optimizing Security Protocol: A Synergy of Bio-inspired Planet Optimization Algorithm with Ensemble Learning-based Attack Detection for Connected and Autonomous Vehicles," *IEEE Access*, 2024.

[21] S. Rahman et al., "SYN-GAN: A robust intrusion detection system using GAN-based synthetic data for IoT security," *Internet of Things*, vol. 26, p. 101212, 2024.

[22] P. Kumar, G. P. Gupta, and R. Tripathi, "Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for IoT networks," *Arabian Journal for Science and Engineering*, vol. 46, no. 4, pp. 3749-3778, 2021.

[23] M. Y. Aldarwbi, A. H. Lashkari, and A. A. Ghorbani, "The sound of intrusion: A novel network intrusion detection system," *Computers and Electrical Engineering*, vol. 104, p. 108455, 2022.

[24] Z. Sun et al., "Optimized machine learning enabled intrusion detection system for internet of medical things," *Franklin Open*, vol. 6, p. 100056, 2024.

[25] M. Vishwakarma and N. Kesswani, "A new two-phase intrusion detection system with Naïve Bayes machine learning for data classification and elliptic envelop method for anomaly detection," *Decision Analytics Journal*, vol. 7, p. 100233, 2023.

[26] H. O. Zahraa, B. Mirzaei, and A. Darroudi, "An efficient automatic modulation recognition using time–frequency information based on hybrid deep learning and bagging approach," *Knowledge and Information Systems*, vol. 66, no. 4, pp. 2607-2624, 2024.