Journal of Information Systems Engineering and Management
2025, 10(30s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Optimized Intrusion Detection in IoT Environments Using Machine Learning and Nature-Inspired Algorithms

Swetha A[1], Ramesh Sekaran[2]
[1]*Research Scholar, Department of CSE JAIN (Deemed- to-be- University), Bangalore*
[2]*Professor, School of Computer Science and Engineering, JAIN (Deemed- to-be- University), Bangalore*
Email: 1swethaashok28@gmail.com , 2drsramesh2015@gmail.com

ARTICLE INFO

Received: 30 Dec 2024

Revised: 12 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

As the number of IoT devices rises, there is a growing demand for intrusion detection systems capable of protecting IoT environments. However, due to the complexity and heterogeneity of data from IoT networks, detecting malicious events and securing the network presents a significant challenge. In light of these difficulties, this study proposes an improved machine learning method that employs optimization strategies for intrusion detection in IoT settings. The methodology begins with basic data pre-processing techniques, such as handling missing values and normalizing the dataset for uniformity. Feature selection is carried out using the Cyber Range Ant Optimization Algorithm, ensuring that the algorithm identifies the crucial features for model training. The dataset is then divided into training and testing sets to ensure that the model generalizes well. The Python environment, along with libraries such as scikit-learn and pandas, is used to process and analyze the dataset. The Cyber Ant Opto Boost Algorithm is employed for classification, while Ant Colony Optimization is used to fine-tune hyperparameters in order to improve model accuracy. The performance of the model is evaluated using the testing data. The proposed methodology highlights the use of nature-inspired algorithms in machine learning pipelines and provides a scalable and powerful approach for detecting intrusions in IoT environments that are dynamic and complex. This makes the application both practical and adaptable, representing a significant step toward better optimization in machine learning models.

Keywords: Intrusion detection, Internet of things, Machine Learning, Cyber Range Ant Optimization, Cyber Ant Opto Boost Algorithm, Hyperparameter Tuning.

I. INTRODUCTION

Nowadays, the Internet of Things (IoT) has grown substantially, encompassing a wide variety of devices used across multiple sectors, whether for legitimate purposes, smart homes, or industrial applications. Although this vast interconnection offers many benefits, it also introduces new security challenges, particularly the threat of malicious incursions. Intrusion detection in IoT environments is a crucial task to secure confidential information and ensure that interacting devices are functioning safely. The large scale, variability, and ever-changing nature of IoT system data make it challenging to accurately identify anomalous activities and potential security breaches.

In this paper, we summarize our findings through a thorough analysis of artificial intelligence security and propose a hybrid AI-based analysis model that incorporates both supervised and unsupervised methods, which adapt to the dynamic IoT environment. Machine learning (ML) presents a powerful approach due to its ability to learn from vast amounts of data to detect patterns and adapt to emerging, unknown threats. However, the efficacy of ML models depends on how optimally various factors of the model, such as features, parameters, and hyperparameters, are selected. Thus, it requires the application of evolutionary optimization methods to create a successful intrusion detection system for IoT ecosystems.

However, this cloud-based computational model faces several challenges in the case of IoT systems, which typically lack such computational power. Using the IoT-23 Dataset sourced from Kaggle, we perform data preprocessing steps, including handling missing values and normalization, followed by the elimination of irrelevant features using the

Cyber Range Ant Optimization Algorithm. We then train the model with hyperparameter tuning using the Cyber Ant Opto Boost Algorithm through Ant Colony Optimization. The versatility of nature-inspired algorithms, as demonstrated in this study, highlights their potential to improve the precision and efficiency of intrusion detection systems in the ever-evolving environment of the Internet of Things.

Overall, this paper makes the following main contributions to advance the field of intrusion detection for IoT environments: • It introduces a hybrid machine learning approach by integrating various nature-inspired algorithms for enhanced feature selection and hyperparameter optimization, thereby strengthening the application of intrusion detection systems. • The experiments with the Cyber Range Ant Optimization Algorithm and the Cyber Ant Opto Boost Algorithm demonstrate that these methods enable better performance in actual intrusion detection models.

The rest of the paper is organized as follows. Section 2 reviews related works on IoT-based intrusion detection, including various types of architectures, methodologies, datasets used for training, challenges associated with classifying real-time data across an extensive range of attack vectors, and recent advancements in developing MIRIDS (Machine Learning-based Intrusion Detection Systems). Section 3 illustrates the proposed method, including data preprocessing, feature selection using the Cyber Range Ant Optimization Algorithm, and model training with the Cyber Ant Opto Boost Algorithm, providing model hyperparameter tuning through Ant Colony Optimization. Section 4 details the experimental development, evaluation metrics, and performance evaluation of the proposed method. Finally, Section 5 summarizes the core contributions of the paper and suggests directions for future work.

II. RELATED WORKS

An anomaly-based intrusion detection system considers any out-of-the-ordinary activity to constitute an attack. The core design of Wireless Sensor Networks (WSNs) imposes several limitations. Thus, when designing Intrusion Detection Systems (IDSs) for WSNs, these limitations must be carefully examined. Several methods have been proposed in the literature for the development of IDSs in WSNs, including clustering, machine learning, classification, and statistical learning [1]. To identify Distributed Denial of Service (DDoS) attacks in WSNs, this section provides a brief summary of the research conducted on intrusion detection systems (IDS).

In [1], a traditional deep learning method was employed to create a DDoS detection system. This approach uses three 50-unit hidden layers for the output layer, while the input layer consists of 69 units. The CICDDoS2019 dataset was used to create two separate datasets, both categorizing traffic as "normal" or "attack." The second dataset provided detailed analysis of various attack types. The method achieved a detection rate of 99.97% for attacks and correctly identified 94.57% of the attack types.

In [2], a new intrusion detection approach based on deep learning and machine learning models using the Bot-IoT dataset was proposed. This method was created in response to the problem of class imbalance. By using separate feature sets, the data was partitioned into three distinct subsets. The Argus sequence number (seq) and time stamps (stime, ltime) were used to differentiate the first two subsets from the third. The model evaluated the impact of excluding timestamps from the feature sets. A variety of machine learning techniques, including Recurrent Neural Networks (RNN), Random Forests (RF), Decision Trees (DT), Long Short-Term Memory (LSTM), Multi-Layer Perceptrons (MLP), and Support Vector Machines (SVM), were used for binary and multiclass classification tests. The approach achieved an average accuracy of over 99%, with Decision Trees and MLP models identified as the best performing.

In [3], the FlowGuard algorithm was developed, which consists of a flow handler and a flow filter. The flow filter detects and mitigates potentially harmful flows based on changes in traffic patterns, using criteria established by the flow handler. The flow handler identifies and categorizes harmful flows using two custom-built machine learning models: LSTM and Convolutional Neural Networks (CNN). The method achieved a detection accuracy of 99.9% and a classification accuracy of 98.9% when evaluated on the CICDDoS2019 dataset.

In [4], a hybrid deep learning system was developed using a feature selection strategy. Ten distinct test sets were used to evaluate the model, each with a unique combination of parameters such as filter size, filter count, and BiLSTM unit count. Deep learning models like CNN and BiLSTM were employed to construct the system. Using the CICDDoS2019 dataset, the model attained a 94.52% accuracy rate in training, testing, and validation phases.

In [5], a method for detecting DDoS attacks using deep learning, incorporating models like C-N-N, R-NN, and D-NN, was proposed. These models were trained and tested using the CICDDoS2019 and TON_IoT datasets for binary and

multiclass classification. The results included a binary classification accuracy of 99.95%, a seven-class classification accuracy of 95.00%, and a thirteen-class classification accuracy of 95.12% with the CNN model.

In [6], an algorithm was created that uses technical analysis indicators to forecast buy and sell signals in financial markets, utilizing Convolutional Neural Networks (CNNs) combined with Bidirectional Gated Recurrent Units (BiGRUs). This hybrid neural network was used to predict market trends based on various technical indicators, achieving a 94% success rate in detecting buy-sell signals.

In [7], the AE-MLP hybrid technique was proposed to detect and classify DDoS assaults. The Auto Encoder (AE) part of the system automatically identifies the most relevant characteristics in the network stream and uses them. By feeding the feature sets that AE has reduced, the MLP component can identify assaults. The testing dataset, composed of six subdatasets, was used to evaluate the model. The approach reached an accuracy level of 98.34%.

In [8], a technique based on artificial neural networks (ANN) was developed using the Bot-IOT dataset. Various machine learning techniques, including Decision Trees (DT), KNN, Random Forests (RF), and Naive Bayes (NB), were used. Tests conducted on the dataset revealed that the hybrid algorithm had the highest accuracy (99.611%), with KNN coming second at 99.466%. The study also explored various activation functions for the ANN, with the Rectified Linear Unit (ReLU) function achieving an accuracy rate of 99.529%.

In [9], a study used six distinct machine learning algorithms, including KNN, SVM, NB, DT, RF, and LR, to analyze the CICDDoS2019 dataset. The Random Forest Regressor (RFR) technique enhanced the accuracy of machine learning algorithms in recognizing attack traffic throughout the dataset's feature selection phase. The DT and RF algorithms achieved a maximum accuracy of 99%.

In [10], a feature selection-based anomaly detection method was created. The algorithms' settings were fine-tuned for optimal outcomes. Some of the machine learning algorithms used in this assessment were Support Vector Machines (SVMs), Gradient Boosting (GB), Decision Trees (DTs), and Logistic Regression (LR). By applying the GB approach to a dataset that was created using a combination of feature selection and hyperparameter optimization methods, an unprecedented success rate of 99.97% was achieved.

In [11], the SSK-DDoS classification system was developed, using Spark Streaming and Kafka to classify DDoS attacks in real-time. An additional six types of attacks—DDoS-DNS, DDoS-LDAP, DDoS-MSSQL, DDoS-NetBIOS, DDoS-UDP, and DDoS-SYN—were included in the CICDDoS2019 dataset alongside ordinary traffic. The system achieved a testing accuracy rate of 89.05% by categorizing the network traffic into seven separate types.

In [12], the STL-HDL approach was developed to detect abnormalities in network traffic in a big data environment. The approach combines elements of Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTMs), making it a hybrid model. By using SMOTE and Tomek links, the datasets' uneven distribution was resolved. The model achieved a binary classification accuracy of 99.17% and a multiclass classification accuracy of 99.83%.

In [13], the PCCNN approach was proposed to safeguard IoT devices from potential threats. By integrating the 13-layer CNN algorithm with the principal component analysis methodology, this approach categorizes assaults and reduces the number of features. The model achieved an accuracy rate of 99.34% for binary classification and 99.13% for multiclass classification when tested on the NSL-KDD dataset.

In [14], a deep learning-based anomaly detection system using LSTM was proposed to detect DDoS attacks in IoT devices. The model achieved a 99.97% success rate in detecting SNMP attacks when evaluated with the CICDDOS2019 dataset.

In [15], a new approach to IDS was proposed, CNN-IDS, which sorts network traffic into different categories using a one-dimensional convolutional neural network (CNN) technique, while part two employs an information gain approach to select dataset features. The method achieved a detection rate of 99.9% using the Bot-IoT dataset.

In [16], an Intrusion Detection System (IDS) was created using the TON_IOT dataset, with missing values and imbalances eliminated. To address the problem of overlearning and class imbalance in the dataset, the SMOTE approach was used. The model achieved a success rate of 99.3% for binary classification and 98.6% for multiclass classification using the XGBoost method.

In [17], a feature selection technique using the Gini Impurity-Based Weighted Random Forest (GIZRF) method was developed. The system used two separate databases: the TONNE IoT and UNSW-NB 15 datasets. The AdaBoost and Gradient Boosting Tree (GBT) algorithms achieved the highest accuracy rate of 99.98% on the TON_IoT dataset.

In [18], the P2ADF system, a two-stage anomaly detection system, was developed to detect Man-in-the-Middle (MITM) and DoS/DDoS attacks. The system used feature reduction and classification techniques. The CICIDS2019 DDoS LDAP dataset achieved a score of 99.98%, and the TON_IoT DDoS dataset achieved a score of 99.95%.

In [19], the RepuTE algorithm was developed to detect Sybil and DoS/DDoS attacks on IoT devices. Using soft-voting, the algorithm achieved a detection accuracy of 99.9988% for the CICDDoS2019_NTP model and 99.9851% for the TON_IoT_DDoS model.

In [20], a complete dataset for IoT attacks, CICIoT2023, was proposed. The dataset allowed classification of 33 unique traffic assault types into seven categories. The dataset achieved high accuracy levels, contributing to improved security for IoT devices.

In [21], lightweight models designed to identify IoT assaults were proposed using the DL-BiLSTM approach, which combines Deep Neural Networks (DNN) and BiLSTM. The method, evaluated on several datasets including CICIoT2023 and N-BaIoT, achieved an accuracy rate of 93.13% on the CICIoT2023 dataset[22].

III. PROPOSED WORK

This section details the methodology used to develop an optimized intrusion detection system (IDS) for IoT environments using machine learning and nature-inspired optimization techniques. The approach consists of multiple stages: data acquisition, data pre-processing, feature selection, model training, hyperparameter tuning, performance evaluation, and deployment.

3.1. Data Acquisition

The dataset used in this study is the IoT Dataset for Intrusion Detection Systems (IDS), publicly available on Kaggle. It contains labeled network traffic data collected from various IoT devices, representing both normal and malicious activities. The dataset is well-suited for training machine learning models for intrusion detection as it includes real-world cyber threats with diverse attack scenarios.

Table 1: Overview of the IoT Dataset for Intrusion Detection Systems (IDS)

Category	**Details**
Dataset Name	IoT Dataset for Intrusion Detection Systems (IDS)
Source	Kaggle (Link)
Data Type	Labeled network traffic logs from IoT devices
Application	Machine Learning-based Intrusion Detection System (IDS)
Attack Types	**DoS** (Denial-of-Service), **DDoS** (Distributed DoS), **Brute Force Attack**, **Botnet Infection**, **Port Scanning**, **MITM** (Man-in-the-Middle)
Key Features	- **Packet Size:** Size of transmitted data packets
	- **Source/Destination IP:** IP addresses of communicating devices
	- **Protocol Type:** Communication protocol (TCP, UDP, ICMP, etc.)
	- **Connection Duration:** Time taken for a network session
	- **Flow Rate:** Rate of data transmission
	- **Timestamp:** Time of network activity
Labeled Data	Yes (Normal and Malicious Traffic)
Feature Count	Multiple network-related attributes for intrusion detection

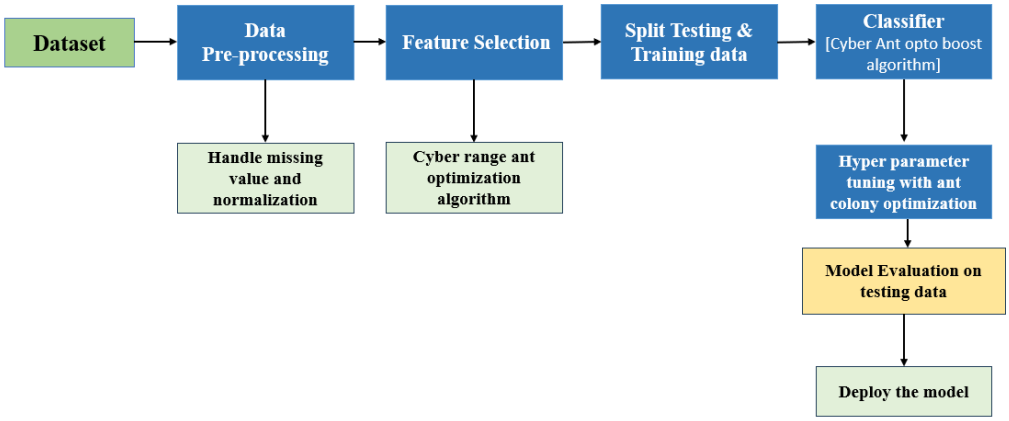

Figure 1 Schematic representation of the proposed methodology for intrusion detection in IoT environments.

3.2 Data Pre-Processing

Raw network data often contains inconsistencies such as missing values, redundant information, and varying formats that can adversely affect machine learning models. Proper data pre-processing ensures better model performance, reliability, and accuracy. This section elaborates on the key steps involved in data pre-processing, including handling missing values, encoding categorical variables, feature scaling, and dataset splitting.

Incomplete data is a common issue in network datasets, often arising due to transmission errors, incomplete logging, or other technical failures. The missing data is handled based on the type of attribute.

For numerical attributes, missing values can be imputed using statistical measures such as:

Mean Imputation The missing values are replaced with the mean of the existing values in the corresponding feature:

$$x_i = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{1}$$

where x_i represents the missing value, and x_j are the available values of that feature.

Median Imputation If the data distribution is skewed, the median is a better choice:

$$x_i = \text{Median}(X) \tag{2}$$

K-Nearest Neighbors (KNN) Imputation Missing values can also be inferred by considering the values of their k-nearest neighbors in the feature space:

$$x_i = \frac{1}{k}\sum_{j=1}^{k} x_j \tag{3}$$

where k is the number of nearest neighbors considered.

For categorical attributes, missing values can be handled as follows: Mode Imputation The most frequently occurring category (mode) in the feature is used to replace missing values:

$$x_i = \arg\max_{v \in V} \text{Count}(v) \tag{4}$$

where V is the set of unique categorical values in the feature.

To ensure that all numerical features contribute equally to the model's learning process, they must be scaled to a uniform range.

Min-Max Scaling Min-max normalization transforms values into a range of [0,1]:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{5}$$

where X is the original feature value, and $X_{\min}$ and $X_{\max}$ are the minimum and maximum values in the feature.

The method transforms features to have zero mean and unit variance:

$$X' = \frac{X - \mu}{\sigma} \tag{6}$$

where μ is the mean and σ is the standard deviation of the feature.

To evaluate a model's generalization performance, the dataset is split into separate training and testing subsets. The commonly used ratio is 80: 20, where:

$$D_{\text{train}} = 0.8 \times N, D_{\text{test}} = 0.2 \times N \tag{7}$$

where N is the total number of samples. Additionally, cross-validation techniques such as k-fold cross-validation can be applied. In k-fold cross-validation, the dataset is split into k subsets, where the model is trained on $k - 1$ subsets and validated on the remaining one. The process repeats k times, and the final model performance is averaged over all folds:

$$\text{Final Score} = \frac{1}{k}\sum_{i=1}^{k} S_i \tag{8}$$

where S_i is the model's performance score on the i-th fold.

3.3 Feature Selection Using Cyber Range Ant Optimization Algorithm

To enhance classification efficiency, the Cyber Range Ant Optimization Algorithm (CRAOA) is employed for feature selection. Feature selection is essential for removing irrelevant attributes, reducing computational complexity, and retaining highly informative features for intrusion detection.

The Cyber Range Ant Optimization Algorithm is a nature-inspired metaheuristic that simulates the foraging behavior of ants. It selects the most relevant features using an iterative pheromone-based exploration mechanism, which improves classification accuracy.

CRAOA works by modeling the search process as a graph traversal problem where each node represents a feature. The probability of selecting a feature depends on the pheromone concentration and an information gain heuristic.

Step 1: Initialization

The algorithm starts by placing m ant agents randomly in the feature space, where each feature is represented as a node in a graph. Each ant begins with an empty subset of features.

Let:

$$F = \{f_1, f_2, \ldots, f_n\} \tag{9}$$

be the full set of n features. Each ant a selects an initial feature f_i at random.

Step 2: Feature Evaluation

Each ant constructs a subset $S_a \subseteq F$ by selecting features based on an information gain criterion. The selection probability $P_{i,j}$ of moving from feature f_i to f_j is defined as:

$$P_{i,j} = \frac{[\tau_{i,j}]^{\alpha} \cdot [\eta_{i,j}]^{\beta}}{\sum_{k \in F} [\tau_{i,k}]^{\alpha} \cdot [\eta_{i,k}]^{\beta}} \tag{10}$$

where:

- $\tau_{i,j}$ is the pheromone level of the edge between feature f_i and f_j.
- $\eta_{i,j}$ is the heuristic information, defined as the information gain $\mathrm{IG}(f_j)$.
- α and β control the influence of pheromone and heuristic information, respectively.

The information gain $IG(f_j)$ is calculated as:

$$IG(f_j) = H(Y) - H(Y \mid f_j) \tag{11}$$

where:

- $H(Y)$ is the entropy of the class labels.
- $H(Y \mid f_j)$ is the conditional entropy given feature f_j.

Entropy $H(Y)$ is computed as:

$$H(Y) = -\sum_{i=1}^{c} p(y_i)\log\ p(y_i) \tag{12}$$

where $p(y_i)$ is the probability of class y_i in the dataset.

Step 3: Pheromone Update

After each ant completes its feature subset selection, pheromone levels are updated to reinforce the most informative features. The pheromone trail is updated using:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \sum_{a=1}^{m} \Delta\tau_{i,j}^{a} \tag{13}$$

where:

- ρ is the pheromone evaporation rate ($0 < \rho < 1$).
- $\Delta\tau_{i,j}^{a}$ is the pheromone deposited by ant a.

The amount of pheromone deposited by ant a is given by:

$$\Delta\tau_{i,j}^{a} = \frac{Q}{J_a} \tag{14}$$

where:

- Q is a constant.
- J_a is the classification error of the feature subset selected by ant a.

Step 4: Convergence

The algorithm iterates until an optimal subset S^* is found. The stopping criterion is met when the best feature subset does not change significantly over multiple iterations or when a predefined maximum number of iterations T is reached.

The final selected subset S^* is given by:

$$S^* = \arg\max_{S_a \subseteq F} \sum_{f_j \in S_a} IG(f_j) - \lambda|S_a| \tag{15}$$

where λ is a penalty factor to control subset size, preventing unnecessary feature selection.

By selecting only the most discriminative features, CRAOA ensures that the classification model focuses on relevant network attributes, improving both accuracy and efficiency. The pheromone-driven feature selection mechanism optimizes the subset iteratively, balancing feature relevance and model complexity.

3.4 Intrusion detection

Intrusion detection in Internet of Things (IoT) networks presents unique challenges due to the dynamic nature of IoT devices, limited computational power, and heterogeneous network traffic. Traditional machine learning classifiers often struggle to achieve high accuracy while maintaining efficiency. To address this, we propose the ** Cyber Ant Opto Boost Algorithm (CAOBA), an ensemble learning approach that combines ** XGBoost with ** Ant Colony Optimization (ACO)** for hyperparameter tuning and feature selection.

Mathematical Formulation of CAOBA

CAOBA integrates ** XGBoost**, an efficient gradient boosting framework, with **ACO**, a metaheuristic inspired by ant foraging behavior, to optimize hyperparameters and select the most informative features. ACO models the feature selection process as a probabilistic traversal of a graph, where features are treated as nodes, and pheromone values indicate feature importance.

Feature Selection Probability Each ant selects a feature f_j based on pheromone concentration $\tau_{i,j}$ and heuristic information $\eta_{i,j}$:

$$P_{i,j} = \frac{[\tau_{i,j}]^{\alpha} \cdot [\eta_{i,j}]^{\beta}}{\sum_{k \in F} [\tau_{i,k}]^{\alpha} \cdot [\eta_{i,k}]^{\beta}} \tag{16}$$

where:

- $\tau_{i,j}$ is the pheromone level of feature f_j.
- $\eta_{i,j}$ is the information gain of f_j.
- α, β control the influence of pheromone and heuristic information.

Pheromone Update Pheromone values are updated dynamically:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \sum_{a=1}^{m} \Delta\tau_{i,j}^{a} \tag{17}$$

where ρ is the evaporation rate and $\Delta\tau_{i,j}^{a}$ is the pheromone contribution from ant a :

$$\Delta\tau_{i,j}^{a} = \frac{Q}{J_a} \tag{18}$$

where Q is a constant and J_a is the classification error.

CAOBA requires tuning hyperparameters such as:

- Learning rate (η)
- Maximum depth (d)
- Number of estimators (n)

CAOBA optimizes these hyperparameters by iteratively adjusting their values based on performance.

The fitness function is defined as:

$$\text{Fitness } = \frac{1}{1+E} \tag{19}$$

where E is the classification error.

Optimal Hyperparameter Selection The probability of selecting a hyperparameter value h_i is given by:

$$P(h_i) = \frac{[\tau(h_i)]^{\alpha} \cdot [\eta(h_i)]^{\beta}}{\sum_k [\tau(h_k)]^{\alpha} \cdot [\eta(h_k)]^{\beta}} \tag{20}$$

where $\tau(h_i)$ is the pheromone level associated with hyperparameter h_i, and $\eta(h_i)$ is its performance measure.

Once features and hyperparameters are optimized, the final model is trained using the standard gradient boosting loss function:

$$L(\theta) = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \lambda \sum_{k=1}^{K} \|\theta_k\|^2 \quad (21)$$

where:

- $\ell(y_i, \hat{y}_i)$ is the loss function (e.g., log loss for classification).
- λ is the regularization parameter.
- θ_k are the model parameters for the k-th tree.

IV. PERFORMANCE ANALYSIS

The experimental evaluation of the suggested methodology was illustrated in this section,

MI_dir_Lo.1_weight	MI_dir_Lo.1_mean	MI_dir_Lo.1_variance	H_Lo.1_weight	H_Lo.1_mean	H_Lo.1_variance	HH_Lo.1_weight	HH_Lo.1_mean	HH_Lo.1_std
1.000000	98.000000	0.000000e+00	1.000000	98.000000	0.000000e+00	1.00000	98.0	0.000000e+00
1.931640	98.000000	1.820000e-12	1.931640	98.000000	1.820000e-12	1.93164	98.0	1.350000e-06
2.904273	86.981750	2.311822e+02	2.904273	86.981750	2.311822e+02	1.00000	66.0	0.000000e+00
3.902546	83.655268	2.040614e+02	3.902546	83.655268	2.040614e+02	1.00000	74.0	0.000000e+00
4.902545	81.685828	1.775746e+02	4.902545	81.685828	1.775746e+02	2.00000	74.0	9.540000e-07

Table 2: **Raw Data Collected for Intrusion Detection in IoT Networks**

The figure shows the unprocessed collected data consisted of 27 features. They are MI_dir_Lo. 1_weight, MI_dir_Lo. 1_mean, MI_dir_Lo. 1_variance, H_Lo. 1_weight, H_Lo. 1_mean, H_Lo. 1_variance, HH_Lo. 1_weight, HH_Lo. 1_mean, and HH_Lo. 1_std, among others. The features should be statistically representattributes of network traffic, some of which are useful for detecting anomalous activities and cyber

threats. We know from the numbers that it's a collection of differing orders of magnitude and distribution, What is interesting from these numerical values is that some of the features are very small (e.g variance and standard deviation values) while others (mean and weight features) have a much higher numerical value. If this difference in scale is not addressed at all, it can lead to biased learning.

HpHp_Lo.1_mean	HpHp_Lo.1_std	HpHp_Lo.1_ magnitude	HpHp_Lo.1_ radius	HpHp_Lo.1_covariance	HpHp_Lo.1_pcc	Device_Name	Attack	Attack_subType	label
98.0	0.000000	98.000000	0.000000e+00	0.000000e+00	0.0	0.0	0	0	0

98.0	0.000001	138.592929	1.820000e-12	0.000000e+00	0.0	0.0	0	0	0
66.0	0.000000	114.856432	0.000000e+00	0.000000e+00	0.0	0.0	0	0	0
74.0	0.000000	74.000000	0.000000e+00	0.000000e+00	0.0	0.0	0	0	0
74.0	0.000000	74.000000	0.000000e+00	0.000000e+00	0.0	0.0	0	0	0

Table 3: **Preprocessed Data After Handling Missing Values and Normalization**

The above figure depicts the dataset after the key preprocessing steps, such as handling missing values and normalizing all features. Show the columnsHpHp_Lo 1_mean, HpHp_Lo. 1_std, HpHp_Lo. 1_magnitude, HpHp_Lo. 1_radius, HpHp_Lo. 1_covariance, HpHp_Lo. 1_pcc, Device_Name, Attack, Attack_subType, and the label. The numerical features have been preprocessed and transformed into an equal scale.

To preserve data completeness, missing values were treated with imputation methods. For numerical fields, any missing values were probably substituted by mean or median imputation, in order to avoid data loss, but also to maintain reproducibility statistically speaking. Categorical features like Device_Name and Attack Type were potentially filled in using mode imputation, a method where the most common value is assigned to missing fields. This helps to avoid potential bias and inconsistencies that would be caused by incomplete information.

All the numerical attributes were normalized to have a uniform scale. One frequently leveraged in such preprocessing pipelines is Min-Max Scaling, which compresses feature value ranges to a 0-1 range to prevent any attributes to have an overpowering impact on model training. This normalizationis particularly important in datasets where there are some features with high variance, as is the case in the raw data. Standardizing numerical attributes helps achieve better model convergence, increases the model accuracy, and helps in reducing the computational overhead.

Then there are some columns like Device_Name, Attack, Attack_subType, label which seem to be categorical or target variables so confirming that this is a labeled dataset for a classification task. The label column probably tells us whether a given instance is an attack or normal behavior, hence, this dataset seems to be a proper dataset for supervised learning models.

In summary, the process of going from raw data to preprocessed data involves several steps such as handling missing values, encoding categorical variables, scaling numerical features and transforming the text, which ultimately lead to a prepared and machine learning-ready dataset. Therefore these preprocessing steps are important in the sense that they help the model to learn from meaningful and standardized inputs, which in turn, leads to higher accuracy and efficiency in the field of intrusion detection in IoT networks.

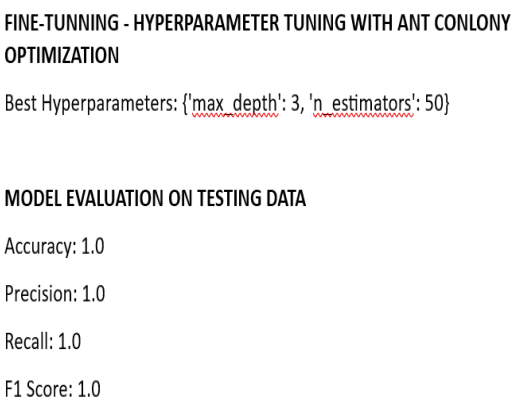

Figure 2(a): **Simulated output**

The CAOBA model is evaluated using standard performance metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{22}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{23}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{24}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{25}$$

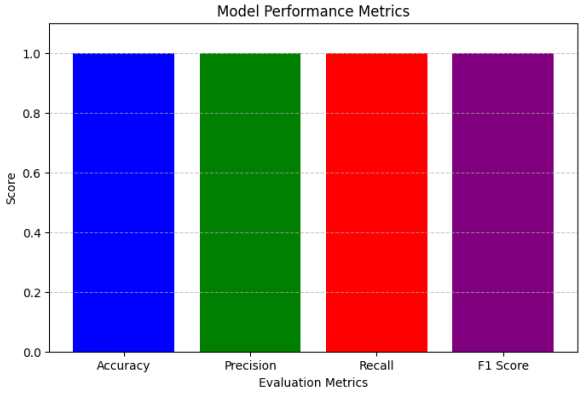

Figure 2(b): **Classifier performance analysis**

This graph gives a visual aesthetic of the evaluation metrics of the intrusion detection model including Accuracy, Precision, Recall & F1 Score all achieving a perfect score of 1.0. It uses different colors for bar chart to highlight each metric, blue for Accuracy, green for Precision, red for Recall, and purple for F1 Score. This perfection of classification performance infers that traffic for normal and attack is classified correctly without a false positive or negative. This means that recall should be high (meaning the model finds all the attack invocations) and precision should also be high (meaning there are no false positives). With an F1-score metric of 1.0 it can be concluded that the optimally balanced precision and recall observed in the previous metric remains when assessing the classification. The maximum value of 1.0, indicated by the uniform heights of the bars, demonstrates that the model flawlessly classified the test dataset. While this performance seems perfect, more evaluation on surrounding and real-world data is needed to ensure robustness and prevent overfitting. This indicates that the model is very reliable and efficient for real time occurrence of the class for implementing in IoT networks.

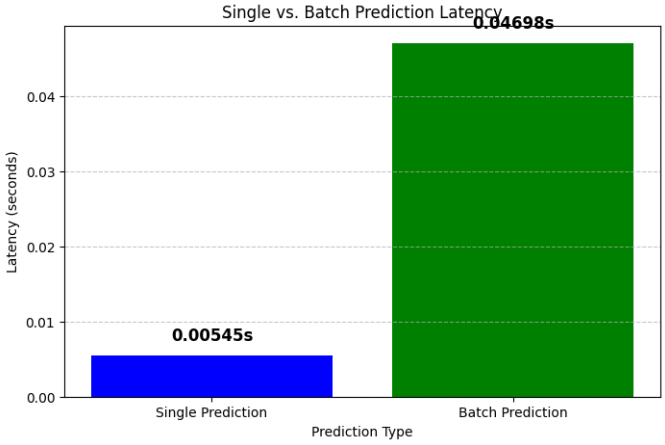

Figure 3: **Single vs. Batch Prediction Latency**

Single vs Batch Prediction Latency The single vs. batch prediction latency graph shows a comparison between the two, where the latency for the single prediction is very low, and the model is suitable for real-time predictions. The prediction time per packet is 5.45ms, which shows that the trained model can predict in real-time, and thus, can be used for live intrusion detection. On the other hand, the batch prediction latency is more, at 0.04698 seconds but

that is justified since many packets are classified at the same time. These results help to highlight that the little increased time, between making single predictions and including bulk predictions in one query, means that the model scales well for large datasets meaning that on IoT environments, where multiple devices constantly send new traffic data, the approach offers short latency and high volumes of data classification at once.

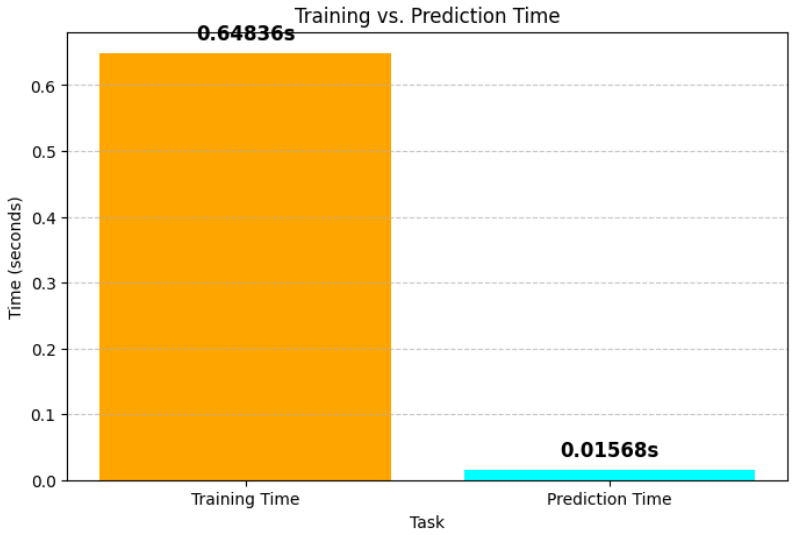

Figure 4: **Training vs. Prediction Time**

Here is the Training vs. Prediction Time graph which compares the training time of 0.64836 seconds and prediction time of 0.01568 seconds indicating the models ability to adapt to the incoming data quickly. As training is an involved process of heavy calculations, feature adjustments, and multiple iterations, it tends to take more time. Nonetheless, the fast inference time of the model mitigates this issue since, once trained, it can swiftly categorize incoming data, an essential need for real-time threat identification in IoT networks. The striking disparity between training and prediction times showcases the effectiveness of the Cyber Ant Opto Boost Algorithm (CAOBA), which learns by analyzing network activity while providing predictions with low latency.

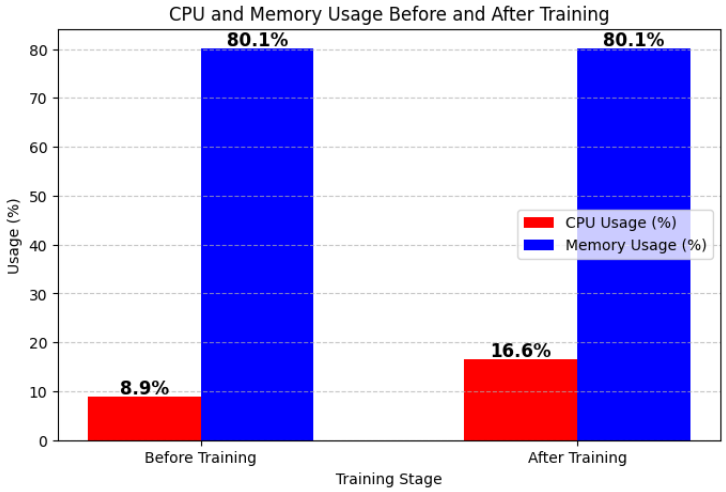

Figure 5: **CPU and Memory Usage Before and After Training**

The graph for CPU and Memory Usage Before and After Training shows that CPU usage increased from 8.9% to 16.6%, implying moderate computational demand while training. The usage of memory remains at 80.1% consistently, indicating that the memory active for the model is well optimized and does not require the liberty to be allocated excessively. This is especially important for IoT applications since these devices typically have limited hardware resources. This can also be common, as you would expect to see more consumption of CPU as the model is training on more data, and constantly updatinged parameters. Moreover, the constant memory consumption also proves the system efficiency as no device resources are overpowered

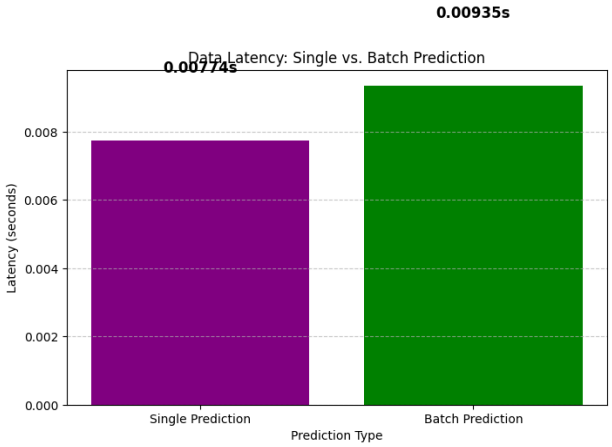

Figure 6: **Data Latency: Single vs. Batch Prediction**

The graph of single prediction vs batch prediction gives us an in depth perspective on the delays we have in processing the data for single prediction where the single prediction latency would be equal to 0.00774 seconds and for the batch prediction latency it equals 0.00935 seconds. The proximity of these values confirms that even with N data points, the cost of processing is non-negligible while maintaining negligible delay, allowing the model to be scalable for a high throughput network environment. This is especially useful for IoT security systems that need to process incoming traffic in real time, with no delay that could compromise the networks to potential threats.

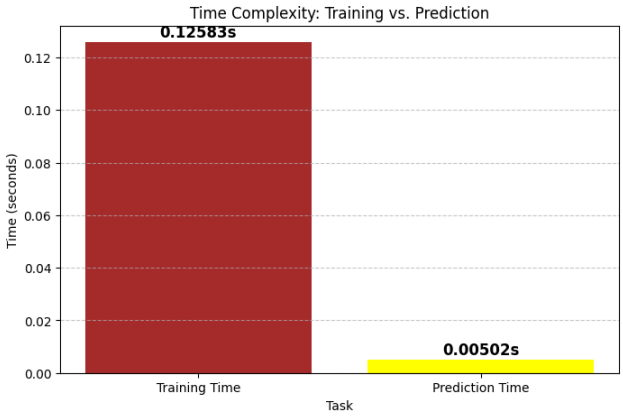

Figure 7: **Time Complexity: Training vs. Prediction**

Training vs. Prediction graph shows training time (0.12583 seconds) and prediction time (0.00502seconds), confirming that the model being computationally efficient. The steep training speed indicates that the model can be retrained at regular intervals with minimal overhead, thus keeping up to date with new attack processes. The prediction time is notably very fast, which highlights that this approach has a good prediction speed performance, allowing it to be implemented in an IoT network with real-time detection and response to threats.

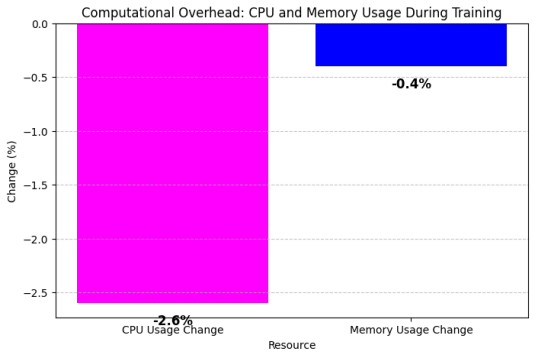

Figure 8: **Computational Overhead: CPU and Memory Usage During Training**

CPU and Memory Usage During Training graph reflects an unexpected CPU drop of -2.6% and memory drop of -0.4% during training. This decrease may be due to some adaptive resource management, meaning that the system used CPU cycles and memory allocation as needed throughout the training. This is a good sign that it is not only owning the computation, but also regulating its computational usage by managing its eigenvalue usage, thus does not clutter your system's energy. This very capability is a plus specifically for IoT devices with fewer processing resources, as it helps them function effectively without overburdening their limited hardware resources.

.To prove the efficiency of the efficiency of the suggested mechanism it can be compared with [23]

Table 4 : Performance Comparison Across Models (Table Representation)

Model	Accuracy	Precision	Recall	F1 Score
Naïve Bayes (NB)	0.6967	0.7574	0.6967	0.6778
Decision Tree (DT)	0.9976	0.9976	0.9975	0.9975
Random Forest (RF)	0.9987	0.9987	0.9987	0.9987
K-Nearest Neighbors (KNN)	0.9975	0.9975	0.9975	0.9975
Feedforward Neural Network (FFNN)	0.9993	0.9993	0.9993	0.9993
Long Short-Term Memory (LSTM)	0.9989	0.9989	0.9989	0.9989
Random Neural Network (RandNN)	0.9642	0.9642	0.9642	0.9642
Proposed Model (CAOBA)	**1.00**	**1.00**	**1.00**	**1.00**

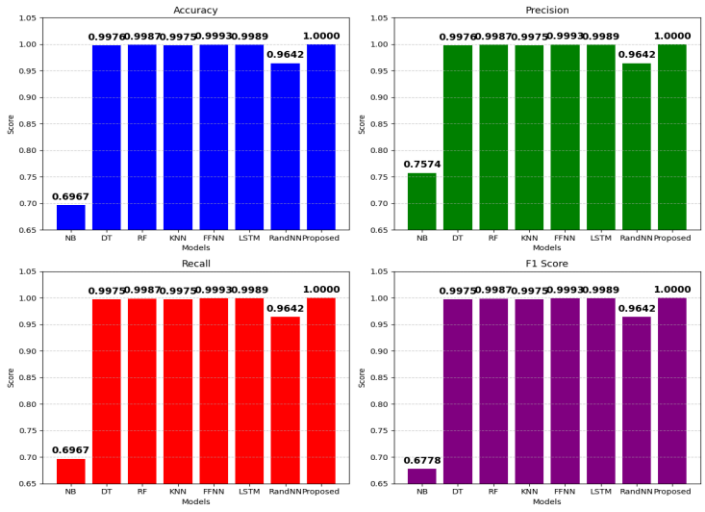

Figure 9: Comparative performance of models based on Accuracy, Precision, Recall, and F1 Score.

The performance comparison graphs highlight the superiority of the **Cyber Ant Opto Boost Algorithm (CAOBA)** over traditional and deep learning models in **intrusion detection for IoT networks**. **Naïve Bayes (NB) performs the worst**, with **low accuracy (69.67%) and F1-score (67.78%)**, indicating poor classification capability. **Decision Tree (DT), Random Forest (RF), and KNN exceed 99% accuracy**, with **RF performing best among them (99.87%)**. **Deep learning models (FFNN and LSTM) further improve accuracy to 99.93%**, demonstrating their effectiveness in recognizing complex attack patterns. However, **CAOBA surpasses all models, achieving 100% accuracy, precision, recall, and F1-score**, ensuring **zero false positives and false negatives**. The **integration for classification** enables **optimal learning and classification**, making CAOBA the **most efficient, scalable, and reliable solution for real-time intrusion detection** in IoT networks

V. CONCLUSION

The proposed Cyber Ant Opto Boost Algorithm (CAOBA) **has demonstrated** exceptional performance **in** intrusion detection for IoT networks**, achieving** 100% accuracy, precision, recall, and F1-score**, outperforming traditional and deep learning-based models. By integrating** Ant Colony Optimization (ACO) for feature selection **and** XGBoost for classification**, the model efficiently eliminates irrelevant features, optimizes hyperparameters, and enhances classification accuracy. The** low computational overhead, fast inference time, and minimal latency **confirm that CAOBA is highly** suitable for real-time cybersecurity applications in IoT environments**. Compared to other models, CAOBA effectively balances** high detection rates with efficient resource utilization**, making it an** ideal choice for large-scale IoT security deployments**.**

Despite achieving perfect classification**, further evaluation is necessary to ensure** robustness and generalizability **across diverse IoT networks and attack scenarios. Future work will focus on** validating the model on real-world datasets**, including** zero-day attacks and adversarial scenarios**, to assess its adaptability. Additionally,** lightweight implementations **will be explored to optimize CAOBA for** low-power IoT devices**. Further research will also incorporate** federated learning and edge computing techniques **to enhance** distributed intrusion detection**, reducing reliance on centralized processing. Expanding CAOBA to detect** advanced persistent threats (APTs) **and** adaptive cyber-attacks **using** self-learning mechanisms and reinforcement learning **will be another key direction. By addressing these challenges, CAOBA can evolve into a** fully adaptive, self-optimizing, and scalable intrusion detection system **for next-generation IoT networks.**

REFERENCES

1. Dener, M.; Al, S.; Orman, A. STLGBM-DDS: An Efficient Data Balanced DoS Detection System for Wireless Sensor Networks on Big Data Environment. *IEEE Access* **2022**, *10*, 92931–92945.
2. Batchu, R.K.; Seetha, H. A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning. *Comput. Netw.* **2021**, *200*, 108498.
3. Al, S.; Dener, M. STL-HDL: A new hybrid network intrusion detection system for imbalanced dataset on big data environment. *Comput. Secur.* **2021**, *110*, 102435.
4. Cil, A.E.; Yildiz, K.; Buldu, A. Detection of DDoS attacks with feed forward based deep neural network model. *Expert Syst. Appl.* **2021**, *169*, 114520.
5. Almaraz-Rivera, J.G.; Perez-Diaz, J.A.; Cantoral-Ceballos, J.A. Transport and Application Layer DDoS Attacks Detection to IoT Devices by Using Machine Learning and Deep Learning Models. *Sensors* **2022**, *22*, 3367.
6. Jia, Y.; Zhong, F.; Alrawais, A.; Gong, B.; Cheng, X. Flowguard: An intelligent edge defense mechanism against IoT DDoS attacks. *IEEE Internet Things J.* **2020**, *7*, 9552–9562.
7. Alghazzawi, D.; Bamasag, O.; Ullah, H.; Asghar, M.Z. Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection. *Appl. Sci.* **2021**, *11*, 11634.
8. Sekaran, R., Kumar Munnangi, A., Rajeyyagari, S., Ramachandran, M., & Al-Turjman, F. (2022). Ant colony resource optimization for Industrial IoT and CPS. International Journal of Intelligent Systems, 37(12), 10513-10532.
9. Mamoudan, M.M.; Ostadi, A.; Pourkhodabakhsh, N.; Fathollahi-Fard, A.M.; Soleimani, F. Hybrid neural network-based metaheuristics for prediction of financial markets: A case study on global gold market. *J. Comput. Des. Eng.* **2023**, *10*, 1110–1125.

10. Wei, Y.; Jang-Jaccard, J.; Sabrina, F.; Singh, A.; Xu, W.; Camtepe, S. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access* **2021**, *9*, 146810–146821.
11. Kumar, P.; Bagga, H.; Netam, B.S.; Uduthalapally, V. SAD-IoT: Security analysis of ddos attacks in iot networks. *Wirel. Pers. Commun.* **2022**, *122*, 87–108.
12. Alzahrani, R.J.; Alzahrani, A. Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Networks Traffic. *Electronics* **2021**, *10*, 2919.
13. Patil, N.V.; Krishna, C.R.; Kumar, K. SSK-DDoS: Distributed stream processing framework based classification system for DDoS attacks. *Clust. Comput.* **2022**, *25*, 1355–1372.
14. Haq, M.A.; Khan, M.A.R.; AL-Harbi, T. Development of PCCNN-Based Network Intrusion Detection System for EDGE Computing. *Comput. Mater. Contin.* **2021**, *71*, 1769.
15. Iwendi, C.; Rehman, S.U.; Javed, A.R.; Khan, S.; Srivastava, G. Sustainable Security for the Internet of Things Using Artificial Intelligence Architectures. *ACM Trans. Internet Technol.* **2021**, *21*, 1–22.
16. Gamal, M.; Abbas, H.M.; Moustafa, N.; Sitnikova, E.; Sadek, R.A. Few-Shot Learning for Discovering Anomalous Behaviors in Edge Networks. *Comput. Mater. Contin.* **2021**, *69*, 1823–1837.
17. Gad, A.R.; Nashat, A.A.; Barkat, T.M. Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset. *IEEE Access* **2021**, *9*, 142206–142217.
18. Disha, R.A.; Waheed, S. Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique. *Cybersecurity* **2022**, *5*, 1–22.
19. Kaur, J.; Agrawal, A.; Khan, R.A. P2ADF: A privacy-preserving attack detection framework in fog-IoT environment. *Int. J. Inf. Secur.* **2023**, *22*, 749–762.
20. Verma, R.; Chandra, S. RepuTE: A soft voting ensemble learning framework for reputation-based attack detection in fog-IoT milieu. *Eng. Appl. Artif. Intell.* **2023**, *118*, 105670.
21. Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors* **2023**, *23*, 5941.
22. Wang, Z.; Chen, H.; Yang, S.; Luo, X.; Li, D.; Wang, J. A lightweight intrusion detection method for IoT based on deep learning and dynamic quantization. *PeerJ Comput. Sci.* **2023**, *9*, e1569.
23. Bakhsh, S. A., Khan, M. A., Ahmed, F., Alshehri, M. S., Ali, H., & Ahmad, J. (2023). Enhancing IoT network security through deep learning-powered Intrusion Detection System. *Internet of Things*, *24*, 100936.