Abid *et al. Journal of Cloud Computing* (2025) 14:72
https://doi.org/10.1186/s13677-025-00785-2

Journal of Cloud Computing

RESEARCH Open Access

A novel IoT threat detection using GWO feature selection and CNN-enhanced LightGBM

Tarek Abid[1], Ahmed Ahmim[1], Faiz Maazouzi[1], Djalel Chefrour[1], Insaf Ullah[2*], Marwa Ahmim[3] and Reham Almukhlifi[4]

Abstract

The rush to deploy IoT devices has greatly increased the threat of cyberattacks. Although prevention methods enhance security, they remain insufficient. Intrusion Detection Systems (IDS) represent a crucial complementary line of defense for IoT networks. In this paper, we propose a novel IDS for the Internet of Things (IoT) that combines the Grey Wolf Optimization (GWO) meta-heuristic, the Light Gradient Boosting Machine (LightGBM) model, and Convolutional Neural Networks (CNN). On the one hand, GWO reduces the number of selected features to the most relevant ones, which positively impacts the computation time in IoT. On the other hand, LightGBM presents the advantage of fast training with low memory usage and performs low latency, whereas CNN performs as a second deep feature extractor of LightGBM outputs and acts as the final classifier. The experimental evaluation of our new model, conducted on the CICIoT2023 and CICIoMT2024 datasets, demonstrated its high performance. For the CICIoT2023 dataset, our model achieved notable performance improvements, with an accuracy of 95.24%, a precision of 95.22%, and a very high true positive rate for several attack classes, such as Distributed Denial of Service, reaching 99.85%. Similarly, for the CICIoMT2024 dataset, our model achieved even higher results, with an Accuracy of 99.50%, a Precision of 99.52%, an F1-Score of 99.51%, an Average Accuracy of 93.22%, and an Average Detection Rate (DR) of 92.36%. Moreover, our model provides a very low false alarm rate with 1.30% and 2.45% for CICIoT2023 and CICIoMT2024, respectively. Therefore, it outperforms well-known machine learning techniques (RF, SVM) and deep learning models, namely DNN, CNN, LSTM, and Multi-head Attention.

Keywords Intrusion detection, IDS, IoT, GWO, LightGBM, CNN, CICIoT2023, CICIoMT2024, Machine learning, Deep learning, Meta-heuristic, Hybrid model

*Correspondence:
Insaf Ullah
Insaf.ullah@essex.ac.uk
[1]Department of Computer Science, Mohamed-Cherif Messaadia University, Street, Souk-Ahras 41000, Algeria
[2]Institute for Analytics and Data Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
[3]Networks and Systems Laboratory, Department of Computer Science, Badji Mokhtar University, Annaba 23000, Algeria
[4]Cybersecurity Department, College of Computer Science and Engineering, Taibah University, Medina 42353, Saudi Arabia

Introduction

The remarkable technological advances of the last decade have made the different electronic devices of our world highly connected to the Internet. This high connectivity offered us many advantages in our daily lives, but also exposed our appliances, information systems, and privacy to a huge number of cyberthreats. Traditional firewalls and antivirus software became insufficient for effectively detecting malicious activities in computer networks [1]. Hackers rely on undisclosed vulnerabilities to access digital systems and launch various attacks, which can lead to the disclosure of very sensitive information. To protect our devices from all these risks, Intrusion Detection Systems (IDS) have emerged as essential defense tools, as they monitor network activities to identify potential threats in real time [2]. They can be classified into two categories: Signature-based IDS (SIDS), effective in detecting previously known attacks, while Anomaly-based IDSs (AIDS) are more suitable for detecting new types of attacks for which no specific signature has yet been identified [3].

In the context of the Internet of Things (IoT), AIDs are more suitable for the detection of abnormal activities that do not correspond to established behavioral patterns [4]. This is due to the high variety of devices and the wide heterogeneity of network technologies employed in these environments. The deployment of an IDS in an IoT environment must consider various scenarios related to its layered architecture, which includes cloud, fog, and edge nodes, as illustrated in Fig. 1. The Edge layer contains devices such as autonomous cars, smart cameras, intelligent industrial machines, etc. The IDS at this layer monitors the network traffic in real-time to detect and block attacks locally. In the intermediate Fog layer, data is processed and analyzed close to the source, to reduce the latency. At this level, the IDS can aggregate and handle data coming from multiple edges. Finally, IDS tasks, which are computationally intensive and require large-scale data, such as machine learning techniques, can be hosted in the cloud [5].

Machine learning techniques play an important role in IDS, thanks to their ability to differentiate legitimate network traffic from malicious attacks and to classify the latter into well-known categories. For instance, a multitude of algorithms have been proposed to tackle the difficult problem of feature selection [6]. Feature selection has a direct impact on Artificial Intelligence (AI) models' performance and accuracy. In this work, we rely on such models in the context of IoT and use optimization meta-heuristics [7] to improve the IDS detection rate and reduction of false alarms. More specifically, we propose a novel IoT-based IDS that combines Grey Wolf Optimization (GWO) algorithm with the Light Gradient Boosting Machine (LightGBM) model and Convolutional Neural Networks (CNN). The goal of this combination is to improve the performance of the IDS model in IoT and increase its detection accuracy.

The main contributions of this paper are:

- The reduction of the dimensionality of IoT traffic data by selecting only the most relevant features. This is achieved by employing the GWO algorithm.
- We implement a hybrid classification system for IoT security, where LightGBM is used for fast, memory-efficient processing and initial classification. Its outputs are then fed into a CNN to further refine feature representation and enhance classification performance.
- The assessment of the model performance in terms of precision and efficiency. We show that

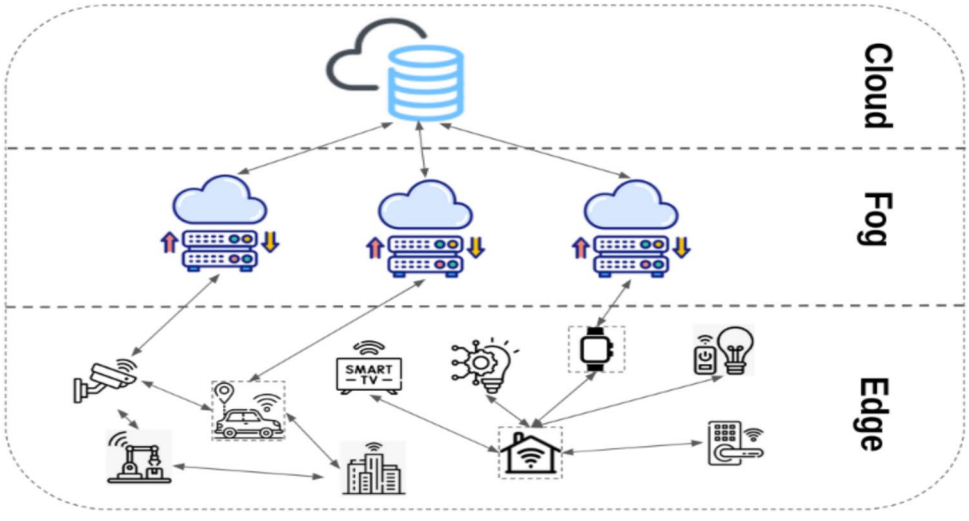


Fig. 1 The layered architecture of IoT

the reduction of features by GWO combined with the LightGBM architecture and CNN model not only improves precision but also efficiency in classification, compared to traditional models.

The remaining sections of this paper are organized as follows: Related works section summarizes related works, Background section presents a background on the different techniques used in this work, The proposed model section details our model, Experimentation section illustrates the experimental evaluation of our IDS discusses the obtained results. At last, we draw our conclusions and list future perspectives in Conclusion section.

Related works

We review hereafter several recent and advanced studies that address various aspects of intrusion detection by combining meta-heuristic techniques and machine learning models in the context of IoT (Table 1).

Ramu et al. [8] implemented a dimensionality reduction approach using Principal Component Analysis (PCA) followed by the Grey Wolf Optimizer (GWO). Attacks were classified using K-Nearest Neighbor, Naïve Bayes, Random Forest, Support Vector Machine, and Deep Neural Networks, yielding accuracies of 99.8%, 86.2%, 93.8%, 97.2%, and 99.9%, respectively.

Similarly, [9] introduced a hybrid intrusion detection method using an unsupervised Sparse Auto-Encoder (SAE) to extract critical features, followed by a Deep Neural Network (DNN) to classify attacks into nine categories. Evaluated on KDDCup'99, NSL-KDD, and UNSW-NB15 datasets, the model achieved a notable accuracy of 99.98% on UNSW-NB15.

Then, [10] presented a feature selection method for intrusion detection combining the Tabu Search algorithm with Random Forest. Tabu Search identified relevant features, and Random Forest served as the learning model. Evaluated on the UNSW-NB15 dataset, the model achieved a detection accuracy of 83.12%.

Moreover, [11] used the PSO-LightGBM approach for feature selection, then used the single-output SVM algorithm to identify malicious data. The model is examined at the UNSW-NB15 dataset with an accuracy rate of 99.80%.

In the same context, [12] proposed an intrusion detection system that mixes the two optimization algorithms, specifically Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO), which selects the best subset of the characteristics. Subsequently, the extracted features are fed to a Random Forest (RF) classifier. The KDDCup99, NSL-KDD, and CICIDS-2017 databases are used to measure the performance of the model, with an average accuracy rate of 99.66% for multi-class classification.

As a continuation, [13] utilized a genetic algorithm for feature selection, reducing the NSL-KDD dataset from 41 to 20 features by eliminating irrelevant ones. Meta-heuristic algorithms (GWO, PSO, MVO, BAT) optimized the selected features, and a hybrid logistic regression (LR) and decision tree (DT) classifier was applied. GWO outperformed other algorithms with an accuracy of 99.44%. However, the genetic algorithm's high computational cost and slow convergence increased the complexity of the IDS.

For detecting denial-of-service cyberattacks in cloud environments, [14] combined the Crow Search Algorithm (CSA) with Recurrent Neural Networks (RNN). CSA, inspired by crow foraging behavior, selected relevant features, and RNN classified the attacks. Tested on the KDDCup'99 dataset, the method achieved a detection accuracy of 94.12%.

Following the same idea, [15] utilized Particle Swarm Optimization (PSO) for feature selection and a Deep Belief Network (DBN) to classify attacks into five categories (Normal, Probe, DoS, R2L, U2R). Tested on the

Table 1 Related works

Ref	Year	Detection method	Feature selection	Dataset	Accuracy
[8]	2020	KNN, RF, SVM, DNN	PCA + GWO	KDD Cup 99	99.90%
[9]	2021	DNN	Sparse Autoencoder	UNSW-NB15	99.98%
[10]	2021	RF	Tabu search	UNSW-NB15	83.12%
[11]	2021	SVM	PSO-LightGBM	UNSW-NB15	99.80%
[12]	2021	RF	PSO, GWO	KDD Cup 99	99.66%
[13]	2022	LR + DT	GA, GWO, PSO, MVO and BAT	NSL-KDD	99.44%
[14]	2022	RNN	Crow Search Algorithm	KDD Cup 99	94.12%
[15]	2022	DBN	PSO	DARPA 1999	96.5%
[16]	2022	Auto-encoder	GWO	NSL-KDD, BoT-IoT	99.9%, 99.9%
[17]	2023	SVM	Chi-square, PCC, MI	ToN-IoT	99.48%
[18]	2023	AdaBoost, RF	BGSA, BGWO	UNSW-NB15	99.41%
[19]	2023	BiLSTM	Harris Hawk optimization	IoT-23, UNSW-NB15	97.34%, 98.12%
[20]	2023	GA-FR-CNN	AAFSO	UNSW-NB, BoT-IoT	94.48%, 94.49%
[21]	2024	DRNN	DFWAFS	BoT-IoT	96.11%
[22]	2024	M-MultiSVM	NGO	CSE-CIC-IDS 2018	99.89%
[23]	2024	DT, RF, LR	GA, PSO, GWO	NSL-KDD	99.58%
[24]	2024	ANN	CVS	CIC IDS-201, BoT-IoT	99.77%, 99.3%
[25]	2024	LSTM	MGA	BoT-IoT, UNSW-NB, N-BaIoT	99.41%, 99.41%, 99.41%

DARPA 1999 dataset, the model achieved an accuracy of 96.5%.

Additionally, [16] developed a two-step intrusion detection model using GWO for feature selection and an auto-encoder for classifying attacks into binary or multiple classes. Evaluated on NSL-KDD and BoT-IoT datasets, the model achieved a detection accuracy of 99.9% for both classification types.

Besides that, [17] proposed a hybrid approach combining filtering methods (Chi-square, PCC, MI) with a meta-heuristic technique. Initially, filtering methods reduce the feature set to k features, which are then binary-encoded for guided population initialization. The SVM algorithm is applied for classification, achieving an accuracy of 99.48% on the ToN-IoT dataset.

Subsequently, [18] proposed a hybrid model for cyber-attack detection using Binary Gravitational Search Algorithm (BGSA) and Binary Grey Wolf Optimization (BGWO) for feature selection, followed by classification with AdaBoost and Random Forest. Evaluated on the UNSW-NB15 dataset, the model achieved an accuracy of 99.41%.

More recently, [19] proposed a hybrid architecture integrating Harris Hawk Optimization for feature selection with a combination of Recurrent Neural Networks (RNN), including Bidirectional Long Short-Term Memory (BiLSTM), Extreme Learning Machine (ELM), and Gated Recurrent Unit (GRU) ensembles. Tested on IoT-23 and UNSW-NB15 datasets, the model achieved detection accuracies of 98.12% and 97.34%, respectively.

After that, [20] introduced the Assimilated Artificial Fish Swarm Optimization (AAFSO) method to identify key features, followed by a Genetic Algorithm and Faster Recurrent Convolutional Neural Network (GA-FR-CNN) for attack classification. The model achieved accuracies of 94.48% and 93.77% on UNSW-NB15 and BoT-IoT datasets, respectively.

Also, [21] proposed an intrusion detection system using the Dynamic Search Fireworks Optimization Algorithm (DFWAFS) for feature selection, followed by a Deep Recurrent Neural Network (DRNN) for attack classification. Validated on the BoT-IoT dataset, the model achieved an accuracy of 96.11%.

Afterward, [22] designed an IDS with a three-part hybrid data optimization approach: feature extraction via Modified Singular Value Decomposition (M-SVD), optimization using Northern Goshawk Optimization (NGO), and classification with a Mud Ring-assisted Multilayer Support Vector Machine (M-MultiSVM). The model achieved accuracies of 99.89% and 97.53% on CSE-CIC-IDS 2018 and UNSW-NB15 datasets, respectively.

To optimize feature selection, [23] employed meta-heuristic algorithms (Genetic Algorithm, PSO, GWO) with machine learning classifiers (Decision Tree, Random Forest, Naïve Gaussian Bayes, Logistic Regression). Tested on NSL-KDD and KDDCup datasets, the best performance was achieved with PSO and Random Forest, yielding an accuracy of 99.57%.

Another relevant study is [24], in which the authors suggested using an ANN for classification and the Chaotic Vortex Search (CVS) meta-heuristic approach for feature selection. The model's accuracy was 99.77% on the CICIDS-2017 dataset and 99.30% on the BoT-IoT dataset.

The authors in [25] combined a Modified Genetic Algorithm model (MGA) and a Long Short-Term Memory network (LSTM). He used MGA for feature selection and GA to determine the optimal hyperparameters of LSTM. The three databases BoT-IoT, UNSW-NB15, and N-BaIoT were used to measure the performance of his model. The model achieved an accuracy score of 99.41%, a detection rate of 99.78%, and a precision score of 98.50%.

Next, [26] proposed an approach for detecting cyber-attacks in industrial Cyber-Physical Systems (CPS-IIoT), integrating privacy-preserving mechanisms and artificial intelligence. The authors combined Pearson correlation coefficient for feature reduction, agglomerative clustering for adaptive network flow grouping, and a BiLSTM model. The test of this model was conducted on two datasets, namely: UNSW-NB15 and X-IIoTID. For both the UNSW-NB15 and X-IIoTID datasets, the model achieved an accuracy score of 99.60% and 99.99% respectively.

In the same context, the authors in [27] present a new framework based on a deep neural network (DNN) for anomaly detection in IoT networks equipped with Cyber-Physical Systems (CPS). This framework also incorporates an advanced explanation method, SHapley Additive exPlanations (SHAP), to enhance its interpretability. Two databases were used, such as Edge-IIoTset and X-IIoTID. For both databases, an accuracy percentage of 100% was achieved for binary classification. For multi-class classification, the Edge-IIoTset database reached 99.98%, while X-IIoTID reached 99.99%.

For intrusion detection in SCADA systems with ISN, the authors in [28] proposed a hybrid ensemble learning model (ELM). Feature extraction from the datasets was performed using principal component analysis (PCA). Bagging, stacking, Adaboost, and a set of Naive Bayes and Support Vector Machine classifiers were optimized using the Gray Wolf Optimizer (GWO). The experiment was conducted in two stages: first, without the use of PCA + GWO for feature extraction and selection on the ELM, and then with PCA + GWO for feature extraction and selection on the ELM. On the UNSW-NB15 dataset, PCA combined with GWO on all NB and SVM classifiers produced an accuracy of 99%, a precision of 100%, a recall rate of 100%, and a detection rate of 99.90%.

Finally, [29] proposed intrusion detection using an autoencoder for feature dimensionality reduction via a deep convolutional neural network (DCNN) and a long short-term memory (LSTM). The LSTM model achieved an accuracy of over 99% and an AUC-ROC of 99.50% on the ICS data, while the DCNN model obtained an accuracy of 96.0% and an AUC-ROC of 97.20% on the gas pipeline network data.

The reviewed studies on intrusion detection for IoT demonstrate a variety of approaches, leveraging meta-heuristic algorithms (e.g., GWO, PSO, CSA), feature selection techniques (e.g., PCA, Tabu Search), and machine learning models (e.g., SVM, RNN, Random Forest), achieving accuracies ranging from 81.5% to 99.98% across datasets like NSL-KDD, UNSW-NB15, and BoT-IoT. However, while some works combined GWO with other methods or used CNN and LightGBM independently, no study has integrated GWO, CNN, and LightGBM together, indicating a potential research gap for enhancing feature selection and classification performance in IoT intrusion detection systems.

Background

This section describes the many machine learning and optimization methods that we employed in our model, initially concentrating on GWO.

Grey Wolf Optimization

Among the most widely used meta-heuristic algorithms, the Grey Wolf Optimizer (GWO) is one of the algorithms that is based on the notion of population, drawing its inspiration from the hierarchical structure and hunting behaviors of grey wolves [30]. Grey wolves live in packs of 5–12 individuals, organized into a strict dominance hierarchy comprising four levels: α, β, δ, and ω, as shown in Fig. 2. The entire set of these four types of wolves constituted the population Ω. The α wolves lead the pack, making decisions on hunting and resting locations. The β wolves assist the α, advising and coordinating pack activities, while dominating lower ranks. The δ wolves, including roles like guards and hunters, submit to α and β but control the ω wolves, the lowest rank, which are subordinate to all others.

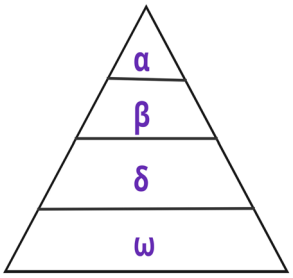


Fig. 2 Hierarchy in the group of the grey wolf

Hunting, as explained in [31], involves tracking and approaching the prey, surrounding it, assailing it until it is immobilized, and then attacking. GWO mimics this behavior to optimize solutions, leveraging the hierarchical roles and cooperative hunting strategies of grey wolves [30].

Mathematical model of GWO

The Grey Wolf Optimizer (GWO) models a pack of 5–12 wolves, each represented by a position vector $\overrightarrow{X}_i = (x_{1,i}, x_{2,i}, \ldots, x_{N,i})$ in an N-dimensional search space. For each dimension j the component $x_{j,i} \in [x_{j,\min}, x_{j,\max}]$, representing the search space bounds or the admissible limits of the decision variable $x_{j,i}$. A fitness function $f(\overrightarrow{X}_i)$ evaluates each wolf's position, identifying the top three solutions as $\overrightarrow{X}_\alpha$ (first best solution), $\overrightarrow{X}_\beta$ (second best solution), and $\overrightarrow{X}_\delta$ (third best solution), while the remaining wolves form the set ω.

The algorithm simulates hunting by updating the positions of ω wolves based on the leaders (α, β, δ). The distance between an ω wolf at $\overrightarrow{X}_i$ and each leader is calculated as in Eqs. (1), (2) and (3).

$$\overrightarrow{D}_{\alpha,i} = \left| \overrightarrow{C}_1 \cdot \overrightarrow{X}_\alpha - \overrightarrow{X}_i \right| \tag{1}$$

$$\overrightarrow{D}_{\beta,i} = \left| \overrightarrow{C}_2 \cdot \overrightarrow{X}_\beta - \overrightarrow{X}_i \right| \tag{2}$$

$$\overrightarrow{D}_{\delta,i} = \left| \overrightarrow{C}_3 \cdot \overrightarrow{X}_\delta - \overrightarrow{X}_i \right| \tag{3}$$

where $\overrightarrow{C}_1, \overrightarrow{C}_2, \overrightarrow{C}_3 = 2\overrightarrow{r}_2$, and $\overrightarrow{r}_2$ is a random vector in $[0, 1]$, introducing randomness to mimic prey encirclement (see Eq. (4)).

$$\overrightarrow{C}_1 = 2\overrightarrow{r}_2 \quad \overrightarrow{C}_2 = 2\overrightarrow{r}_2 \quad \overrightarrow{C}_3 = 2\overrightarrow{r}_2 \tag{4}$$

The new positions of ω wolves are computed relative to the leaders in Eq. (5), (6) and (7).

$$\overrightarrow{X}_{1,i} = \overrightarrow{X}_\alpha - \overrightarrow{A}_1 \cdot \overrightarrow{D}_{\alpha,i} \tag{5}$$

$$\overrightarrow{X}_{2,i} = \overrightarrow{X}_\beta - \overrightarrow{A}_2 \cdot \overrightarrow{D}_{\beta,i} \tag{6}$$

$$\overrightarrow{X}_{3,i} = \overrightarrow{X}_\delta - \overrightarrow{A}_3 \cdot \overrightarrow{D}_{\delta,i} \tag{7}$$

with $\overrightarrow{A}_1, \overrightarrow{A}_2, \overrightarrow{A}_3 = 2\overrightarrow{a} \cdot \overrightarrow{r}_1 - \overrightarrow{a}$, and $\overrightarrow{r}_1 \in [0, 1]$ adds stochasticity (see Eq. (8)).

$$\begin{aligned}\overrightarrow{A}_1 &= 2\overrightarrow{a}\cdot\overrightarrow{r}_1 - \overrightarrow{a}\\ \overrightarrow{A}_2 &= 2\overrightarrow{a}\cdot\overrightarrow{r}_1 - \overrightarrow{a}\\ \overrightarrow{A}_3 &= 2\overrightarrow{a}\cdot\overrightarrow{r}_1 - \overrightarrow{a}\end{aligned} \tag{8}$$

The parameter $\overrightarrow{a}$ controls exploration vs. exploitation, decreasing linearly from 2 to 0 over iterations, as defined in Eq. (9).

$$\overrightarrow{a} = 2 - \frac{2t}{\mathrm{MaxItr}} \tag{9}$$

where t is the current iteration, and MaxItr is the maximum number of iterations. The updated position of each ω wolf is the average of the three influences, as given in Eq. (10).

$$\overrightarrow{X}_i(t+1) = \frac{\overrightarrow{X}_{1,i} + \overrightarrow{X}_{2,i} + \overrightarrow{X}_{3,i}}{3} \tag{10}$$

This iterative process optimizes $\overrightarrow{X}_i$ toward the global optimum by balancing exploration (via $\overrightarrow{A}$ and $\overrightarrow{C}$) and exploitation (via $\overrightarrow{a}$). The whole process of the GWO algorithm is illustrated in Fig. 3. The following subsection outlines how GWO can be employed for feature selection.

Feature selection with GWO

Feature selection (FS) aims to reduce the number of features by retaining the most relevant ones and removing redundant ones, thereby improving model performance and reducing computational complexity [32]. FS methods are classified into three types: filter-based, which uses statistical analysis to identify relevant features [2]; wrapper-based, which selects feature subsets by optimizing an objective function (e.g., accuracy, recall, precision) [10];

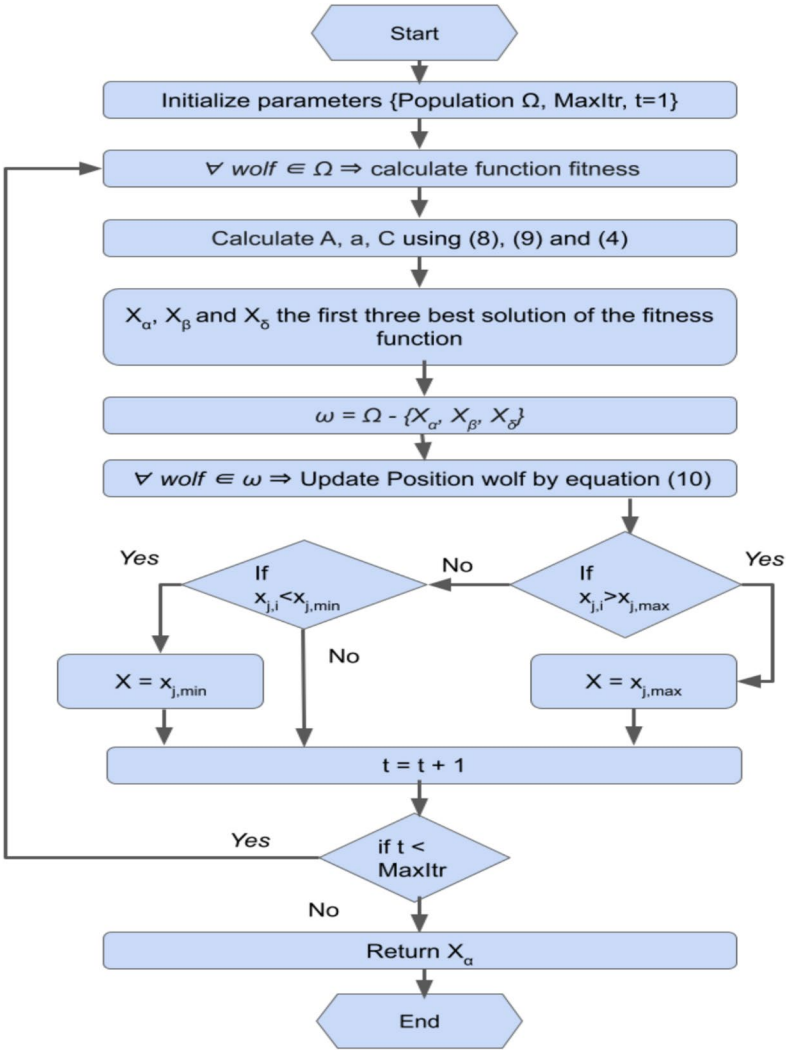


Fig. 3 Hierarchy in the group of the grey wolf

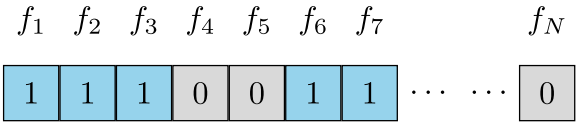


Fig. 4 Binary selection sequence representation

and embedded, where FS occurs during the learning process [2]. This study adopts a wrapper-based FS approach using the GWO to determine an optimal feature subset.

In GWO-based FS, each wolf represents a candidate feature subset, encoded as a binary vector $\vec{X}_i = (x_{1,i}, x_{2,i}, \ldots, x_{N,i})$ of size N (total features), where:

- $x_{j,i} = 1$: Feature is selected.
- $x_{j,i} = 0$: Feature is unselected.

The population (5–12 wolves) is initialized with random binary vectors, as shown in Fig. 4. Each wolf's vector $\vec{X}_i$ is evaluated using a fitness function, typically the accuracy of a learning model (e.g., Decision Tree, SVM), aiming to maximize performance while minimizing the number of selected features.

GWO updates the wolves' positions using Eq. (10). Since the search space is binary (0 or 1), the updated position $\frac{\vec{X}_{1,i}+\vec{X}_{2,i}+\vec{X}_{3,i}}{3}$ is thresholded to determine feature selection:

$$x_{j,i} = \begin{cases} 1, & \text{if } \frac{\vec{X}_{1,i}+\vec{X}_{2,i}+\vec{X}_{3,i}}{3} \geq 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

This iterative process optimizes the feature subset by balancing classification accuracy and feature sparsity.

Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep learning model designed to extract features from input data using convolutional layers [33]. Its architecture, as illustrated in Fig. 5, consists of the following layers:

- Convolutional Layer: Applies N_1 kernels (weight matrices) to the input data, extracting features like textures and edges, producing N_1 feature maps of size $H_1 \times W_2$.
- Pooling Layer: Reduces feature map dimensions while retaining key information, using methods such as Max Pooling, which selects the highest value within a region, or Average Pooling, which calculates the average value across the region.
- Flatten Layer: Converts feature maps into a single vector for input to the fully connected layer.
- Fully Connected Layer: Comprises multiple layers where each neuron connects to all neurons in the next layer, culminating in a softmax layer for classification or regression.

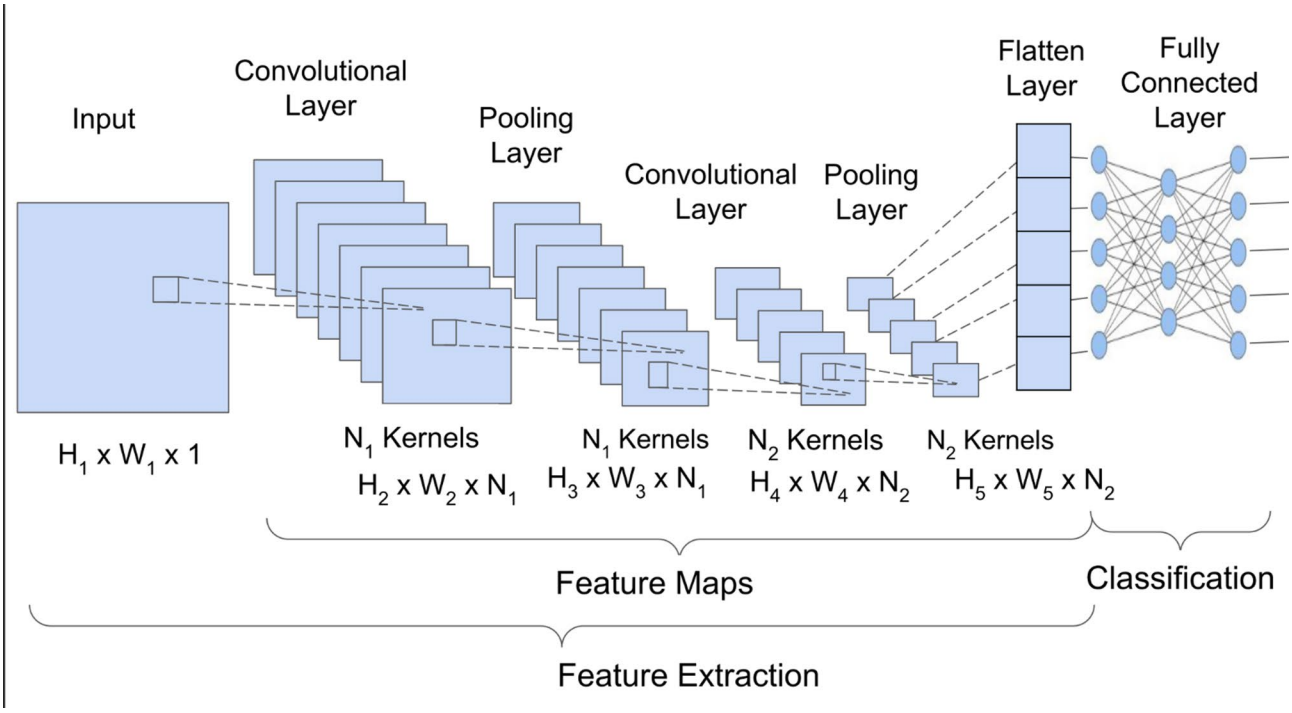


Fig. 5 Architecture of a Convolutional Neural Network (CNN)

Light Gradient Boosting Machine

Light Gradient Boosting Machine (LightGBM), developed by Microsoft, is a fast and efficient machine learning algorithm for classification and regression tasks, particularly on large datasets [34]. It builds on the Gradient Boosting Decision Tree (GBDT) framework, where sequential decision trees correct errors of prior trees, enhancing model performance.

LightGBM employs a leaf-wise tree-building strategy, as shown in Fig. 6, splitting the leaf with the greatest loss reduction, unlike the level-wise approach that adds nodes simultaneously per level. Leaf-wise construction yields higher accuracy and faster convergence but results in unbalanced trees, while level-wise produces balanced trees with lower error reduction.

Tuning LightGBM's parameters manually is complex due to their significant impact on performance. Thus, we use the Optuna library in Python, which leverages Bayesian optimization for efficient hyperparameter tuning, outperforming traditional methods. Key parameters include `objective` (e.g., binary or multiclass), `learning_rate` (controls learning speed), `num_class` (number of classes), `n_estimators` (total trees), `feature_fraction` (random feature sampling per iteration), `bagging_fraction` (random data sampling per iteration), and `boosting_type` (boosting method).

The architecture of LightGBM consists of several successive decision trees; each decision tree is built from several terminal leaves. When an instance or sample traverses a tree based on the decision rules until it reaches in a single determined leaf. The identifier of this leaf is called the leaf index. These indices constitute compact and highly informative features and already highly informed by the data patterns, as they directly encapsulate the patterns learned by the model. We thus obtain a discriminative representation ready to be processed by a deep learning model (CNN, DNN, etc.). Instead of passing raw features (with noise, correlations, and different scales), we pass clean, compact, and more expressive indices than the original input features. Moreover,

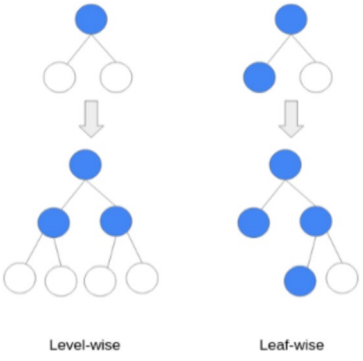


Fig. 6 The generation strategy of the tree in LightGBM

each class contains n estimators trees. For the LightGBM architecture, the total number of trees is: `n_estimators * num_class`.

The proposed model

This section outlines our approach to integrate the LightGBM algorithm with the Convolutional Neural Network (CNN) to develop an efficient and accurate intrusion detection system, tailored for deployment at the Fog computing layer. We start by presenting the overall system architecture, which includes design and deployment considerations. Then, we detail the internal architecture of the hybrid model and provide a comprehensive evaluation of its performance.

The overall system architecture

Our main objective is to develop an intrusion detection system tailored for IoT environments. This system will be deployed on fog nodes, whose primary role is to detect intrusions targeting IoT devices. The process, illustrated in Fig. 7, begins with data collection from various edge nodes, managed by the fog nodes. These fog nodes collect network traffic, analyze it, and filter out relevant information. The filtered data are then transmitted to cloud nodes, where experts re-filter and label the dataset in preparation for model training. Subsequently, GWO is used for feature selection processes. Then, our hybrid model is trained using this new dataset. It is important to note that the training process will be updated periodically to ensure the model remains effective against emerging threats. Once training is complete, the trained hybrid model is distributed to the fog nodes, where it is deployed as an analyzer within the intrusion detection system. After deployment, the system will continuously monitor network activity and detect various types of attacks across the edge nodes.

The proposed hybrid model

The general structure of our model is detailed in Fig. 8. To create this hybrid model, we start using the GWO algorithm for feature selection, which allows us to reduce the number of features while preserving the most essential ones and eliminating those that are unnecessary. After that, we train the first part of our model, namely the LightGBM model, using the selected sub-features and the best parameters calculated by the Optuna function, namely: `n_estimors`, `learning_rate`, `boosting_type`, `lambda_l1`, `lambda_l2`, etc. During the training operation, LightGBM built a certain number of decision trees according to the `n_estimors` parameter. This parameter was set to 600, representing a good compromise between available resources and model accuracy.

The trees are built sequentially, with each new tree correcting the errors of the previous tree by minimizing

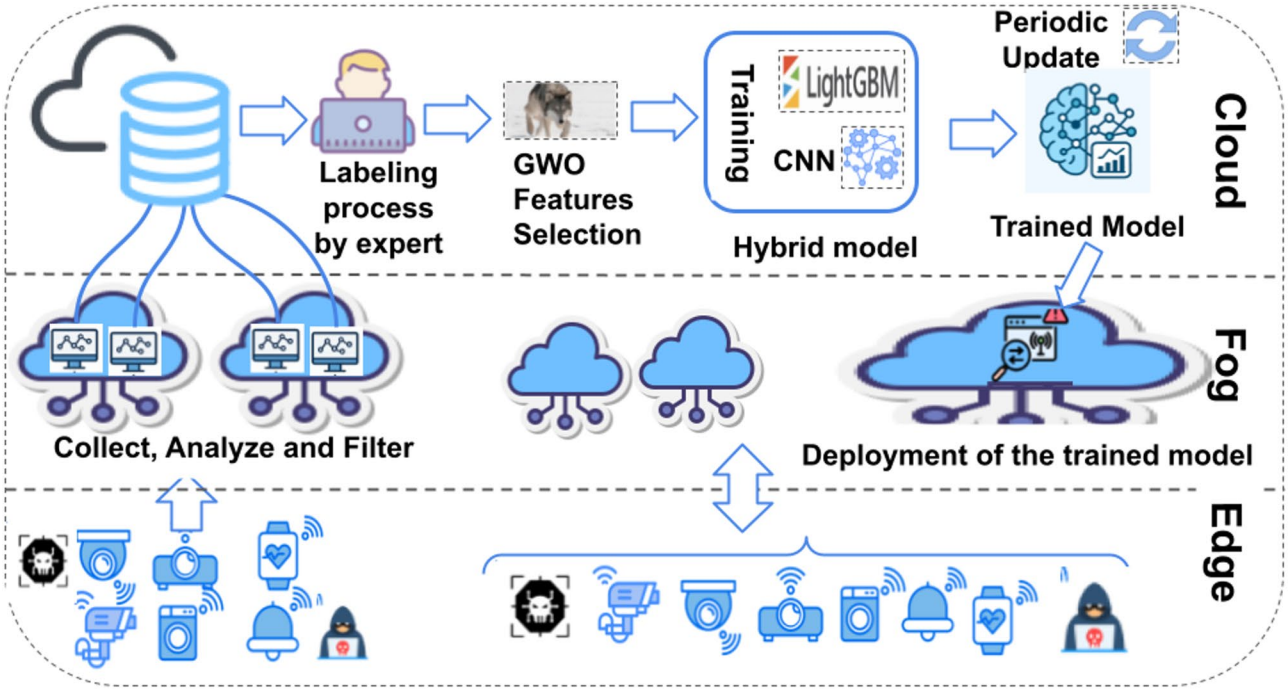


Fig. 7 The overall system architecture

a loss function, specifically the 'multi-logloss' function, which handles classification problems. At the end of the learning operation, LightGBM allows us to retrieve the outputs that represent the indices of the leaves for each sample. The number of these indices is 4800 (corresponding to 600 * 8) indices that are used as discrete features (input). These features are directly injected into the embedding layer to transform them into dense vectors of size 16 that will be used by the CNN model. Next, the outputs of the embedding layer (4800, 16) are connected to the Conv1D layer with 64 filters and a kernel size of 3, as illustrated in Fig. 8. The outputs of the Conv1D layer (4798, 64) are connected to the MaxPooling1D layer, with a pool size of 2. Next, a 30% Dropout layer is applied to prevent overfitting. The outputs of the Dropout layer (2399, 64) are connected to the Flatten layer, which has 153536 output vectors. An additional 40% Dropout is applied to reduce overfitting. Subsequently, the 153536 Dropout outputs are propagated to the Dense layer of 128 neurons with a Relu activation function. Another 50% Dropout layer is used before the output layer. Finally, a dense layer with 8 outputs that represents the classes to be predicted with a Softmax activation function.

Experimentation

The datasets used in this study, namely the CICIoT2023 [35], and CICIoMT2024 [36], exhibit significant class imbalance. To minimize its impact, we retained all samples from minority classes and applied a threshold-based random downsampling strategy to reduce the size of the majority class. This approach mitigated the highly unbalanced training subset and helped reduce bias toward frequently occurring classes. In addition, the overall binary distribution between benign and attack traffic appears relatively balanced. Our primary objective is to distinguish between specific attack types to enable more effective countermeasures. To achieve this, we place particular emphasis on developing a model with strong generalization capabilities, which enables the detection of previously unseen threats in real-world environments. Consequently, we prioritized evaluation metrics that are robust to class imbalance, including the class-wise True Positive Rate (TPR) and Average Accuracy. This ensures a more realistic and comprehensive assessment of the model's effectiveness across all classes.

Figure 9 illustrates the complete experimental processing and modeling pipeline developed using the CICIoT2023 [35] and CICIoMT2024 [36] datasets. The process is divided into multiple crucial phases, for CICIoT2023, beginning with the concatenation of 169 raw CSV files into a single dataset. In the preprocessing phase, we first clean them, and then we apply a random sampling technique to reduce data volume while maintaining class representation. Subsequently, feature normalization is applied to improve model stability and convergence by scaling the input variables to a uniform range. To enable multi-class classification appropriate for intrusion detection tasks, the class labels are then reorganized to represent eight and six different categories for CICIoT2023 and CICIoMT2024, respectively. Afterward,

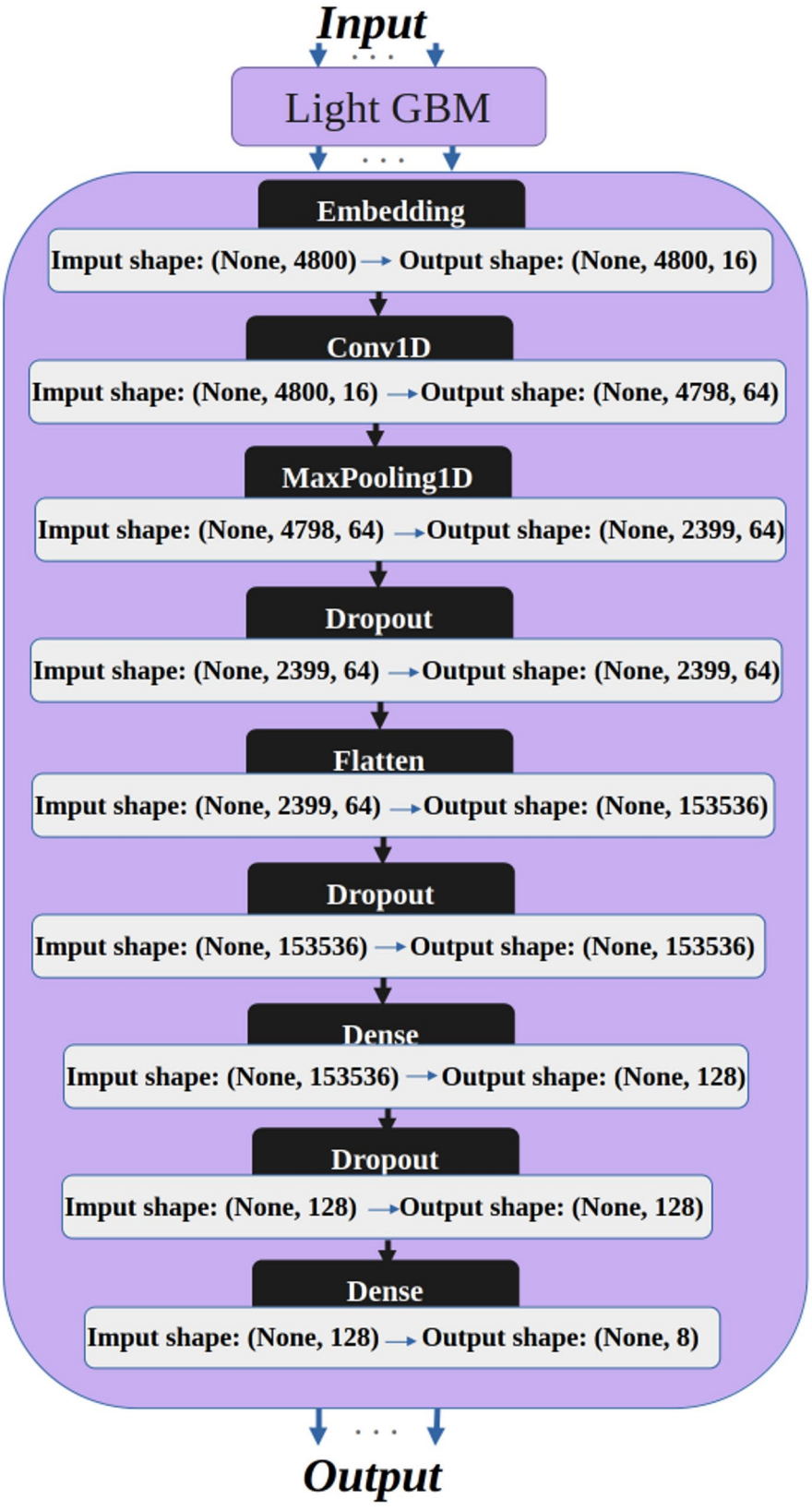


Fig. 8 General architecture of our model

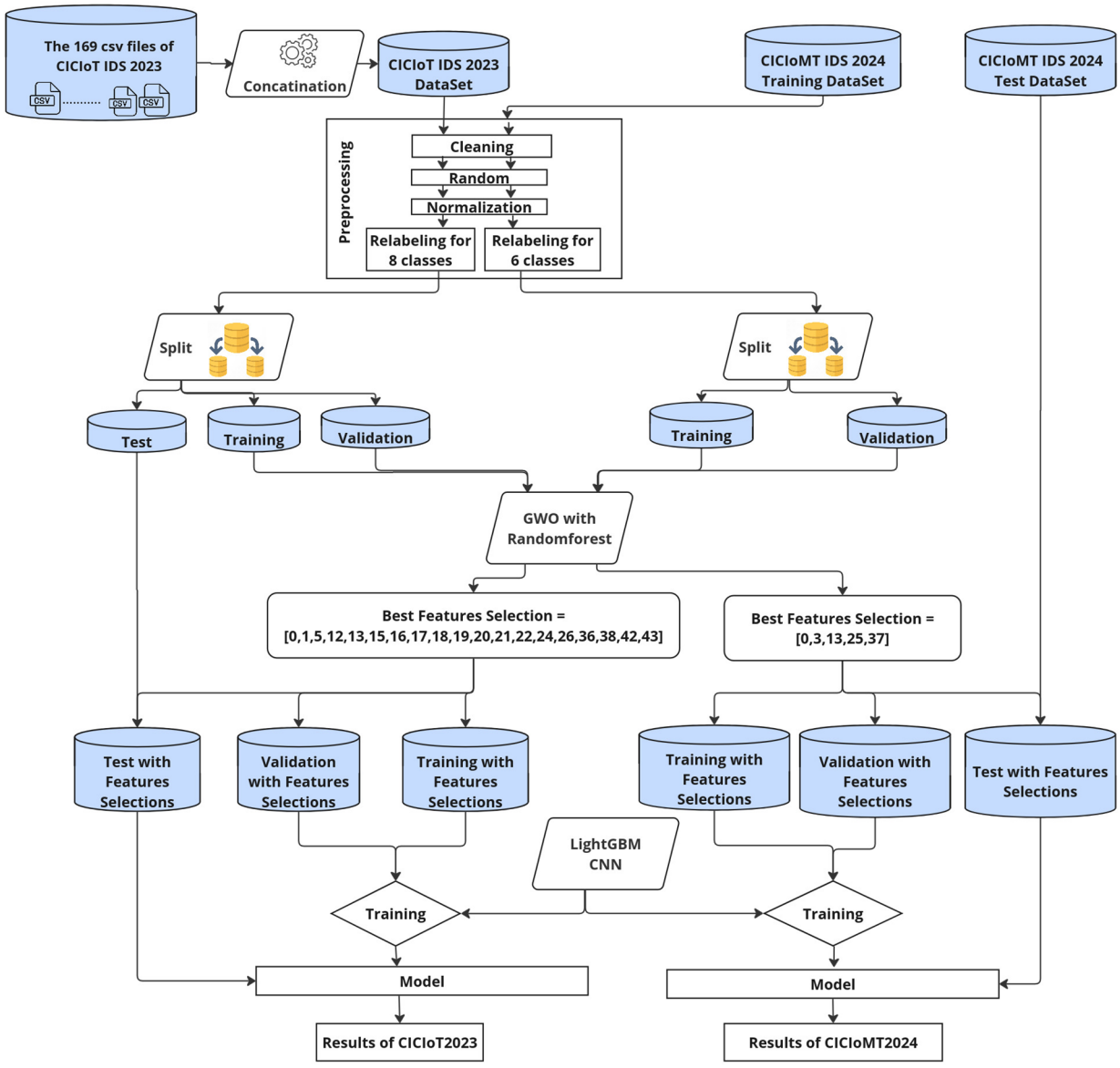


Fig. 9 Different steps to build and test our model

the preprocessed CICIoT2023 dataset is partitioned into three subsets — training, validation, and test — unlike the CICIoMT2024 dataset, where the data is partitioned into only two subsets — training and validation —. These separations constitutes an essential step in the training process.

Next, we apply the Grey Wolf Optimizer (GWO) algorithm to reduce dimensionality and enhance the model's discriminative capacity. Figure 9 shows the best subset of relevant features selected using Random Forest as a classifier. After that, a hybrid classification model combining Light Gradient Boosting Machine (LightGBM) and a Convolutional Neural Network (CNN) is trained using the training subset of the selected features. This hybridization aims to integrate the advantages of gradient boosting and deep learning to provide reliable and accurate intrusion detection. Finally, the trained model is evaluated on the independent test subset to assess its generalization capability. The role of each process will be described in the following paragraphs.

Preparation of raw data

The experimental evaluation is conducted using the CICIOT2023 and CICIoMT2024 datasets, a collection

of data containing various attacks targeting IoT devices, making it easier to develop security analytics applications for real-world IoT operations. The dataset CICIoT2023 lists 33 types of attacks, as shown in Fig. 10 and Table 2, classified into eight distinct categories, as illustrated in Fig. 11 and Table 2: DDoS, DoS, Recon, Web, brute force, Benign, spoofing, and Mirai. In contrast, the dataset CICIoMT2024 contain 18 types of attacks, as presented in Fig. 12 and Table 3, and six different classes, as shown in Fig. 13 and Table 3: DDoS, DoS, Recon, MQTT, Benign, and spoofing. Attacks are carried out by malicious IoT devices targeting other IoT devices. These datasets offer a complete view of the different attacks observed in IoT environments. On the one hand, the total size of CICIoT2023 is 13.75 GB, divided into 169 files in CSV format, and contains 44 features and a label. On the other hand, the CICIoMT2024 dataset is composed of a training set of 1.69 GB and a test set of 377.67 MB in CSV format, with 44 features and a label. These datasets include several attacks that are not available in other IoT datasets [35]. They are accessible on the CIC Dataset website (https://www.unb.ca/cic/datasets/index.html).

Concatenation

The files of the dataset of CICIOT2023 are divided into 169 CSV files, and each file contains 47 features. We concatenated them into a single dataframe using the Pandas library in order to split them into a training subset, a validation subset, and a test subset. This preprocessing step was applied to the CICIoT2023 dataset; however, the CICIoMT2024 dataset is already provided in separate training and test sets, so we divided the training subset into training and validation sets.

Cleaning

In this step, we clean the CICIoT2023 dataset by removing all rows containing missing, NaN, or Inf values. We also remove irrelevant columns—specifically, the Timestamp (ts) feature, as it only records the time of the attack event and does not provide useful information for prediction. Additionally, we discard the DHCP, SMTP, and Telnet columns, since their values remain constantly zero and therefore do not contribute to the learning process. After this cleaning process, we obtain a dataset containing 44 features, as presented in Table 4. Unlike

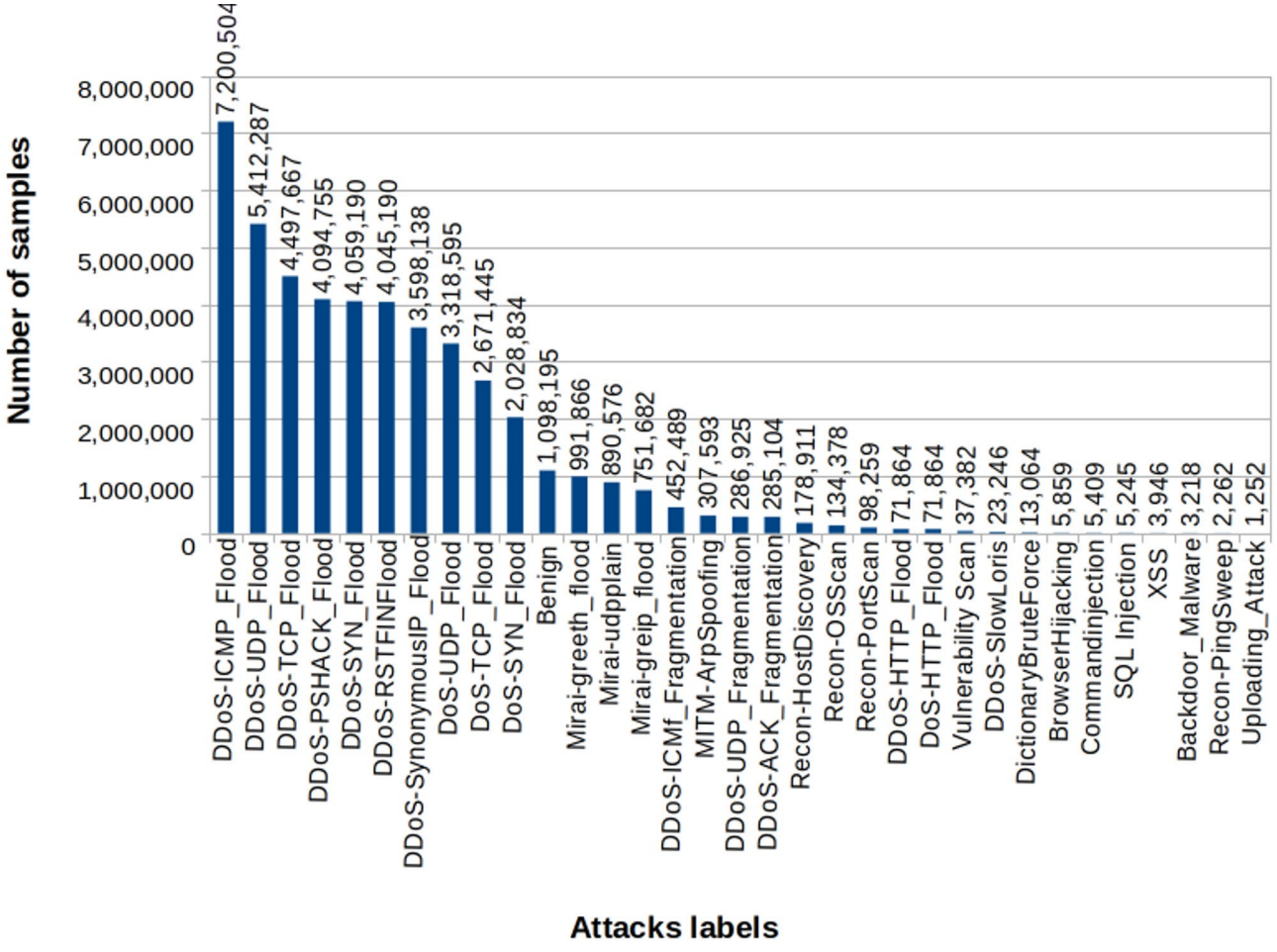


Fig. 10 The 33 types of attacks for CICIoT2023

Table 2 Number of instances for each attack and category for CICIoT2023

Type of connection			All Dataset	Sample	Percent	Train	Val	Test
Bening			1098195	120000	10.93%	86487	9610	23903
Attack (DDoS)	**DDoS**	**ACK Fragmentation**	285104	4000	1.40%	2900	322	778
		UDP Flood	5412287	4000	0.07%	2876	320	804
		SlowLoris	23426	4000	17.07%	2860	317	823
		ICMP Flood	7200504	4000	0.05%	2928	325	747
		RSTFIN Flood	4045285	4000	0.10%	2897	322	781
		PSHACK Flood	4094755	4000	0.10%	2870	319	811
		HTTP Flood	28790	4000	13.89%	2818	313	869
		UDP Fragmentation	286925	4000	1.39%	2878	320	802
		ICMP Fragmentation	452489	4000	0.88%	2869	319	812
		TCP Flood	4497667	4000	0.09%	2927	325	748
		SYN Flood	4059190	4000	0.10%	2882	320	798
		SynonymousIP Flood	3598138	4000	0.11%	2876	319	805
		Total DDoS	33984560	48000	0.14%	34580	3842	9578
Attack (DoS)	**DoS**	**TCP Flood**	2671445	4000	0.14%	2892	321	787
		HTTP Flood	71864	4000	5.57%	2905	323	772
		SYN Flood	2028834	4000	0.20%	2876	320	804
		UDP Flood	3318595	4000	0.12%	2858	317	825
		Total DoS	8090738	16000	0.19%	11531	1281	3188
Attack (Recon)	**Recon**	**Ping Sweep**	2262	2262	100.0%	1628	181	453
		OS Scan	98259	4000	4.07%	2867	318	815
		Vulnerability Scan	37382	4000	10.70%	2864	318	818
		Port Scan	82284	4000	4.86%	2867	318	815
		Host Discovery	134378	4000	2.97%	2847	316	837
		Total Recon	354565	18262	5.15%	13072	1452	3738
Attack (Web-Based)	**Web-Based**	**Sql Injection**	5245	4000	76.26%	2833	315	852
		Command Injection	5409	4000	73.95%	2888	321	791
		Backdoor Malware	3218	3218	100.0%	2321	258	639
		Uploading Attack	1252	1252	100.0%	899	99	254
		XSS	3846	3846	100%	2777	308	761
		Browser Hijacking	5859	4000	68.27%	2907	323	770
		Total Web-Based	24829	20316	81.82%	14625	1624	4067
Attack	**Brute Force**	**Dictionary Brute Force**	13064	4000	30.62%	2906	323	771
Attack (Spoofing)	**Spoofing**	**Arp Spoofing**	307593	4000	1.30%	2868	319	813
		DNS Spoofing	178911	4000	2.23%	2833	315	852
		Total Spoofing	486504	8000	1.64%	5701	634	1665
Attack (Mirai)	**Mirai**	**GREIP Flood**	751682	4000	0.53%	2858	317	825
		Greeth Flood	991866	4000	0.40%	2886	321	793
		UDPPlain	890576	4000	0.45%	2891	321	788
		Total Mirai	2634124	12000	0.45%	8635	959	2406
		Total Attack	45575320	126578	0.27%	91049	10116	25413
Total			46673515	246578	0.52%	177537	19725	49316

CICIoT2023, the CICIoMT2024 dataset is already preprocessed, with no missing, NaN, or Inf values, and all features being consistent and informative—thus eliminating the need for any additional cleaning, as shown in Table 5.

Random selection

The large size of the CICIoT2023 dataset (13.74 GB) makes it almost impossible to run machine learning programs. In contrast, the CICIoMT2024 dataset is more lightweight, with a total size of approximately 2.07 GB (1.69 GB for training and 377.67 MB for testing). This capacity leads to a data storage problem, especially if we have limited resources in terms of RAM and processor speed. To address these issues, we reduced the size of the datasets by randomly selecting samples while maintaining sufficient diversity in attack classes. Specifically, we selected a representative subset of the CICIoT2023 dataset, consisting of 246,578 samples, was selected to reduce computational complexity while preserving the overall

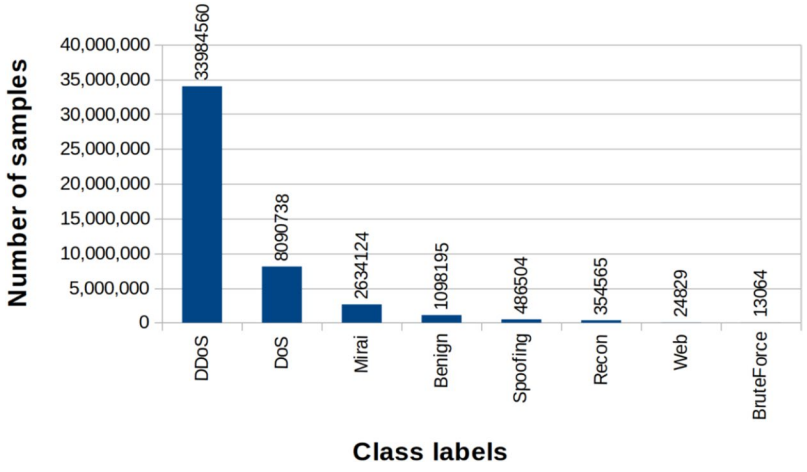


Fig. 11 Number of instances for each class for CICIoT2023

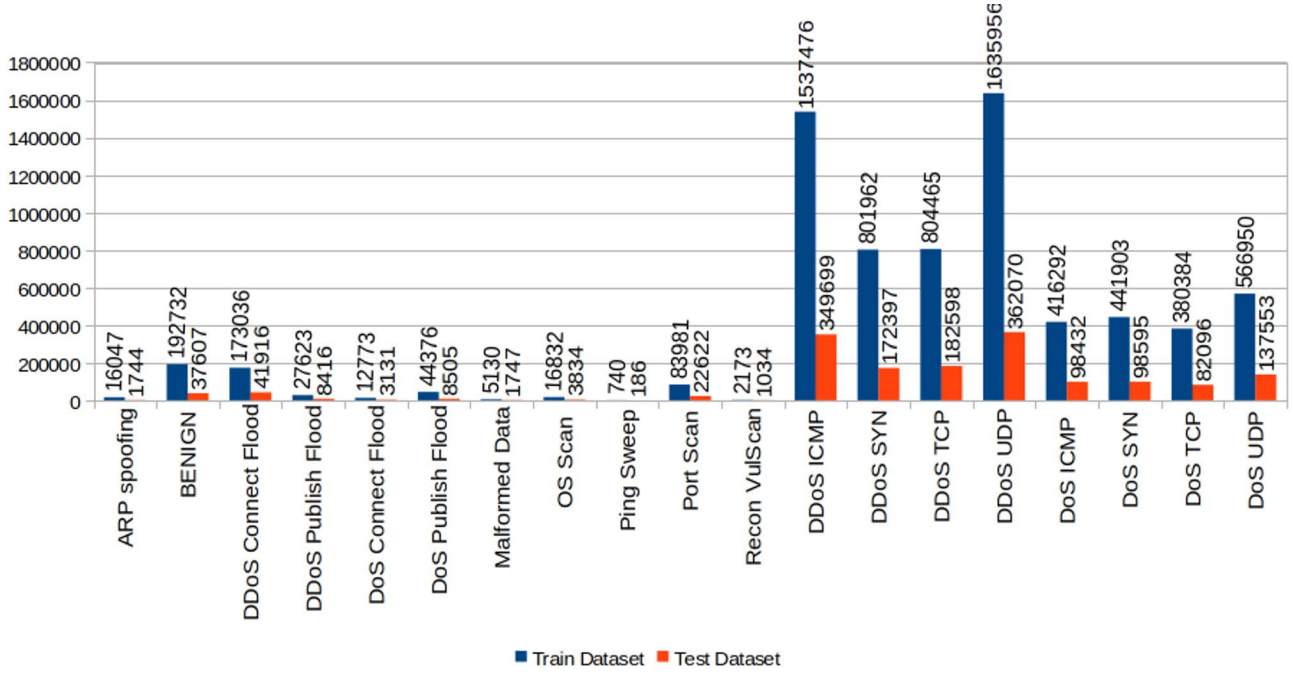


Fig. 12 The 18 types of attacks for CICIoMT2024

class distribution. The Fig. 14 shows the overall distribution of the selected sample. This subset was subsequently divided into training, validation, and test sets following the proportions detailed in Table 2: 177,537 samples (72%) for training, 19,725 samples (8%) for validation, and 49,316 samples (20%) for testing. Concerning the CICIoMT2024 dataset, we randomly selected 222,913 samples from the original training set to create a representative subset, as illustrated in Fig. 15 and specified in Table 3. This subset was divided into 178,330 samples (80%) for training and 44,583 samples (20%) for validation. In contrast to CICIoT2023, the complete original test set of CICIoMT2024 was preserved for evaluation to ensure a comprehensive performance assessment. Figure 15 and Table 3 summarize the detailed distribution of CICIoMT2024 subsets.

Normalization

Our datasets contains data at different scales which are between the minimum value and the maximum value. Normalization or standardization consists of putting all the values between [0, 1] or between [−1, 1], it makes it possible to accelerate and reduce the calculation time and also allows the rapid convergence of the model. There are several variants to normalize data, namely: Min-Max Normalization, Z-Score Normalization, Mean Normalization, etc. For our work, we have chosen the Min-Max Normalization method, which reduces all the values of

Table 3 Number of instances for each attack and category for CICIoMT2024 (The bold text is meaningful; it corresponds to the attack names, the attack classes, and their total.)

Type of connection			All Dataset	Train+Val	Percent	Train	Val	Test
Bening			192732	100000	51.89%	80000	20000	37607
Attack (MQTT)	**MQTT**	**DDoS Connect Flood**	173036	2500	1.44%	2000	500	41916
		DDoS Publish Flood	27623	2500	9.05%	2000	500	8416
		DoS Connect Flood	12773	2500	19.57%	2000	500	3131
		DoS Publish Flood	44376	2500	5.63%	2000	500	8505
		Malformed Data	5130	2500	48.73%	2000	500	1747
		Total MQTT	262938	12500	4.75%	10000	2500	63715
Attack (Recon)	**Recon**	**OS Scan**	16832	2500	14.85%	2000	500	3834
		Ping Sweep	740	740	100%	592	148	186
		Port Scan	83981	2500	2.98%	2000	500	22622
		Recon VulScan	2173	2173	100%	1738	435	1034
		Total Recon	103726	7913	7.62%	6330	1583	27676
Attack (DDoS)	**DDoS**	**DDoS ICMP**	1537476	20000	1.30%	16000	4000	349699
		DDoS SYN	801962	10000	1.24%	8000	2000	172397
		DDoS TCP	804465	10000	1.24%	8000	2000	182598
		DDoS UDP	1635956	20000	1.22%	16000	4000	362070
		Total DDoS	4779859	60000	1.25%	48000	12000	1066764
Attack (DoS)	**DoS**	**DoS ICMP**	416292	10000	2.40%	8000	2000	98432
		DoS SYN	441903	10000	2.26%	8000	2000	98595
		DoS TCP	380384	10000	2.62%	8000	2000	82096
		DoS UDP	566950	10000	1.76%	8000	2000	137553
		Total DoS	1805529	40000	2.21%	32000	8000	416676
Attack (Spoofing)	**Spoofing**	**ARP Spoofing**	16047	2500	15.58%	2000	500	1744
		Total Attack	6968099	122913	1.76%	98330	24583	1576575
Total			7160831	222913	3.11%	178330	44583	1614182

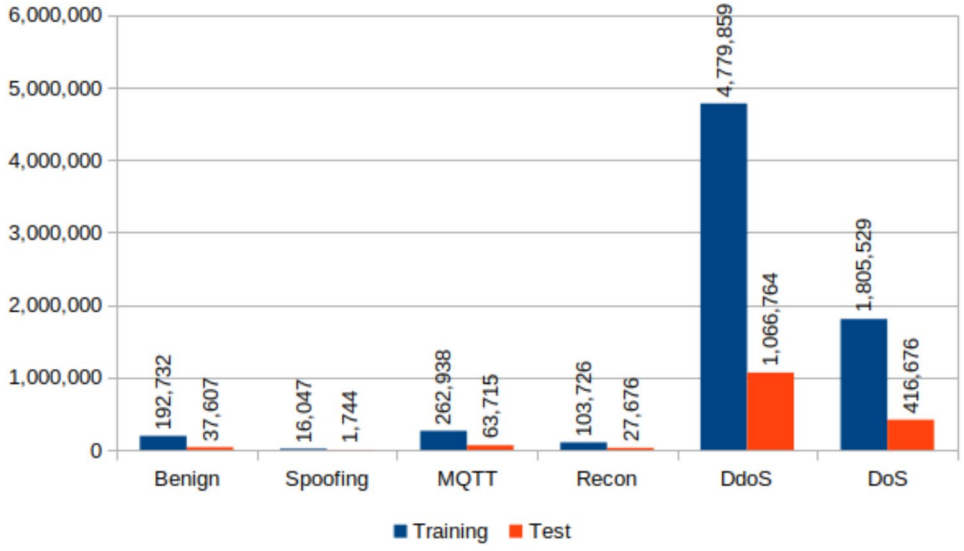


Fig. 13 Number of instances for each class for CICIoMT2024

our database between 0 and 1 (Formula (12)). Where $x_i(j)$ is the data source, min(x(j)) is the minimum value of column j, and max(x(j)) is the maximum value of column j.

$$\overline{x_i(j)} = \frac{x_i(j) - min(x(j))}{max(x(j)) - min(x(j))} \tag{12}$$

Relabeling

The feature named "label" in our datasets represents the nature of the captured traffic. The CICIoT2023 dataset

Table 4 The CIC-IoT-2023 dataset features Names after removing usefulness ones

Feature	Name	Feature	Name
0	Flow duration	22	SSH
1	Header Length	23	IRC
2	Protocol Type	24	TCP
3	Duration	25	UDP
4	Rate	26	ARP
5	Srate	27	ICMP
6	Drate	28	IPv
7	Fin flag number	29	LLC
8	Syn flag number	30	Tot sum
9	Rst flag number	31	Min
10	Psh flag number	32	Max
11	Ack flag number	33	AVG
12	Ece flag number	34	Std
13	Cwr flag number	35	Tot size
14	Ack count	36	IAT
15	Syn count	37	Number
16	Fin count	38	Magnitude
17	Urg count	39	Radius
18	Rst count	40	Covariance
19	HTTP	41	Variance
20	HTTPS	42	Weight
21	DNS	43	Label

Table 5 The CIC-IoMT-2024 dataset features Names after removing usefulness ones

Feature	Name	Feature	Name
0	Header Lengthn	22	IRC
1	Protocol Type	23	TCP
2	Duration	24	UDP
3	Rate	25	DHCP
4	Srate	26	ARP
5	fin flag number	27	ICMP
6	syn flag number	28	IGMP
7	rst flag number	29	IPv
8	psh flag number	30	LLC
9	ack flag number	31	Tot sum
10	ece flag number	32	Min
11	cwr flag number	33	Max
12	ack count	34	Avg
13	syn count	35	Std
14	fin count	36	Tot size
15	rst count	37	IAT
16	HTTP	38	Number
17	HTTPS	39	Magnitue
18	DNS	40	Radius
19	Telnet	41	Covariance
20	SMTP	42	Variance
21	SSH	43	Weight
		44	Label

includes benign traffic and 33 different types of attacks, as indicated in Fig. 10, while the CICIoMT2024 dataset comprises benign traffic and 18 types of attacks, as illustrated in Fig. 12. The names of the attacks are represented as labels. These labels are converted into a digital format to optimize mathematical calculations and make model training faster and more efficient. In the CICIoT2023 dataset, malicious attacks are grouped into seven classes (DDoS: 1, DoS: 2, Mirai: 3, Recon: 4, Spoofing: 5, Web: 6, BruteForce: 7). Additionally, by including benign traffic (Benign: 0), we obtain a total of eight classes. On the other hand, the CICIoMT2024 dataset includes a smaller set of attacks, organized into five classes (Spoofing : 1, MQTT : 2, Recon : 3, DDoS : 4, DoS : 5), along with benign traffic, resulting in a total of six classes.

Data segmentation (split)

We have divided the CICIoT2023 dataset into training, validation, and test subsets based on a Specified ratio. This separation of data ensures a reliable evaluation of our model. Moreover, it allows the model to learn from the training and validation subsets and to be tested using the test subset. In this setup, we allocated 72% of the dataset to the training set (177,537 rows), 8% for the validation set (19,725 rows), and 20% for the testing set (49,316 rows), with a total of 246,578 lines. For the CICIoMT2024 dataset, we selected 222,913 rows as a representative subset for the CICIoMT2024 dataset. In accordance with CICIoT2023, this subset was divided into 80% for training set(178,330 rows) and 20% for the validation set (44,583 rows). The complete test set provided with CICIoMT2024 was preserved in its entirety for the final evaluation. The train-test-split method from the Sklearn model selection module is used to divide the data into training, validation, and testing sets. Finally, we eliminate rows with aberrant values (outliers) from the training and validation subdatasets, which can lead to improved model performance in terms of accuracy, precision, F1-score, and other metrics. We used Isolation Forest, an effective algorithm for outlier detection [37].

GWO

This approach involves selecting a minimum number of features while maximizing the fitness function. At the end of the operation, GWO chooses the best subset of relevant features from the model using Random Forest classifiers as the fitness function. The subset is named BestFeature, which corresponds to the best feature of our model. After running the GWO script, we identified 18 features as the most relevant, as presented in Table 6 among the 43 features of the CICIOT2023 dataset. Similarly, for the CICIoMT2024 dataset, GWO has selected the 5 best sub-feature out of the 44 available, as shown in Table 7.

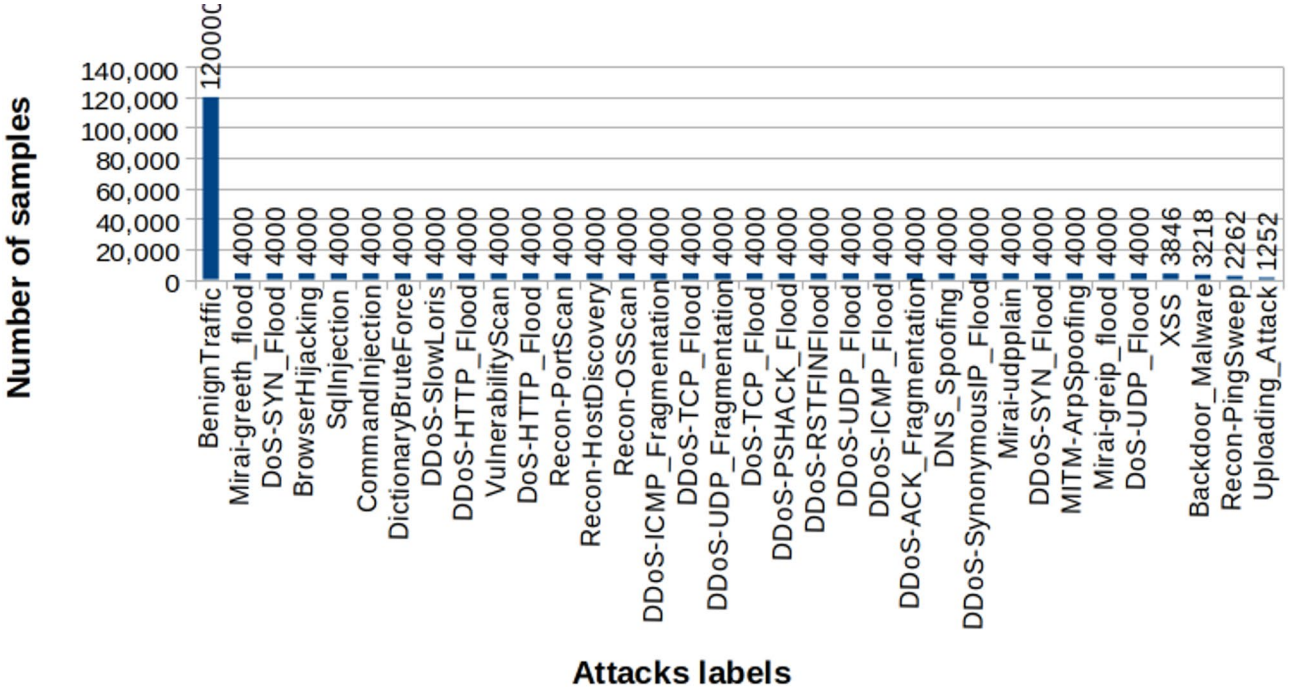


Fig. 14 Overall distribution of the selected sample for CICIoT2023

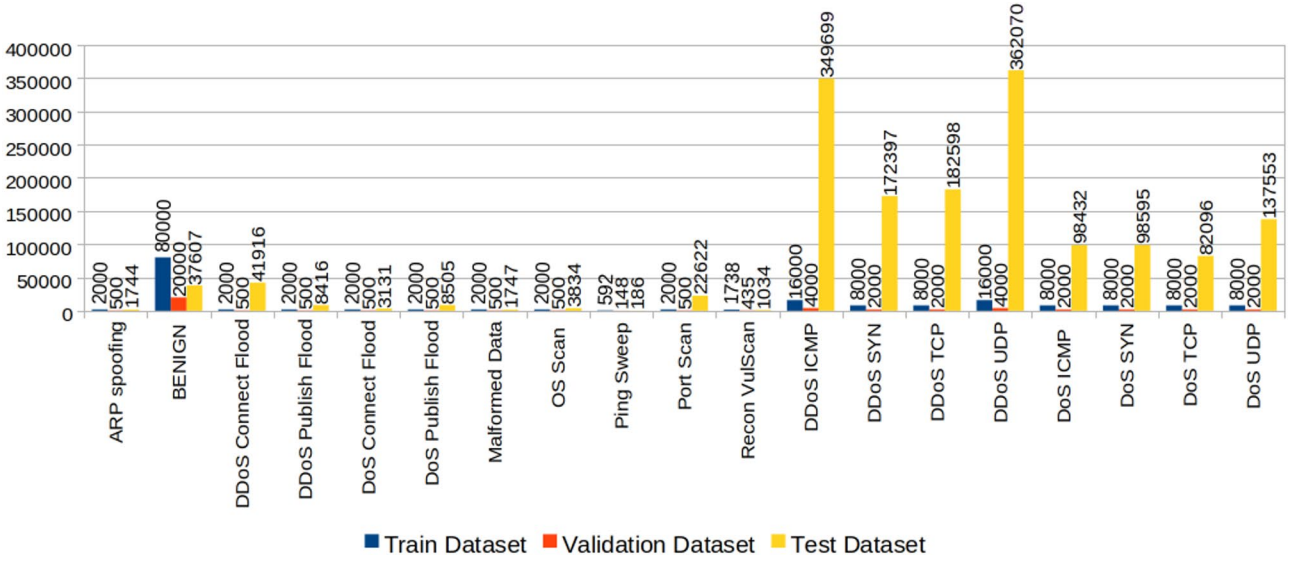


Fig. 15 Overall distribution of the selected sample for CICIoMT2024

Table 6 Best sub-Features using GWO for RF (CICIoT2023)

Parameter	Best sub-Features Selected	Number of selected features
RF	[0,1,5,12,13,15,16,17,18,19,20,21,22,24,26,36,38,42]	18

Table 7 Best sub-Features using GWO for RF (CICIoMT2024)

Parameter	Best sub-Features Selected	Number of selected features
RF	[0,3,13,25,37]	5

Experiment setup

We realized our implementation on the online platform Kaggle, which offers computing resources to train machine learning and deep learning models. Kaggle offers a processing Space of 30 GB of RAM, a disk Space

for storage and processing of 57.6 GB, and an Intel(R) Xeon(R) CPU @ 2.00 GHz processor, which accelerates the execution of complex networks. Kaggle also offers an NVIDIA Tesla P100 GPU with 30 hours of execution per week. We used Python as the programming language and also used libraries such as Tensorflow, Scikit-learn, Pandas, Numpy, Matplotlib, etc.

Performance metrics

To measure system performance and efficiency, we use the confusion matrix that contains four possible values, as indicated in Table 8: True Positives (TP) and True Negatives (TN) which detects whether there is an attack or not (normal) respectively; False Positives (FP) indicates that the model predicts an attack in anticipation of another form of attack by mistake. False Negatives (FN): The model did not report an attack while it was present. An efficient IDS must minimize the rates of FN and FP while maximizing those of TN and TP. Therefore, we use several metrics that give an overview of this objective, namely: Accuracy (Eq. (17)), F1-score (Eq. (14)), Detection Rate (DR) (Eq. (13)), True Positive Rate (TPR) (Eq. (16)), Precision (Eq. (15)), Detection Rate Average (Eq. (18)), False Alarm Rate (19), Accuracy Average ((20)). In these equations, we use C(x) for the connections of type x, including both attack and benign traffic, and A(x) for only the attacks of type x.

$$\mathrm{DR} = \frac{TP}{TP + FN} \tag{13}$$

$$\text{F1-score} = \frac{2.TP}{2.TP + FN + FP} \tag{14}$$

$$\mathrm{Precision} = \frac{TP}{TP + FP} \tag{15}$$

$$TPR_{C(x)} = \frac{TP_{C(x)}}{TP_{C(x)} + FN_{C(x)}} \tag{16}$$

$$\mathrm{Accuracy} = \frac{\sum^{NBclass} \mathrm{TP}}{\sum^{NBclass} (\mathrm{TP} + \mathrm{FP})} \tag{17}$$

$$DR_{Average} = \frac{\sum TPR_{A(x)}}{NB_{OfA(x)}} \tag{18}$$

$$\mathrm{FAR} = 1 - \frac{TP_{Benign}}{TP_{Benign} + FN_{Benign}} \tag{19}$$

$$ACC_{Average} = \frac{1}{NBClass} \sum TPR_{C(x)} \tag{20}$$

Table 8 The confusion matrix

		Predicted	
		Positive	**Negative**
Real	Positive	TP	FN
	Negative	FP	TN

Table 9 Hyperparameter tuning for GWO

Parameter	Value
Number of Wolves	20
Number of Features	43
Number of Iterations	15
Fitness Function	accuracy
Search space	[0, 1]
$\overrightarrow{r}_1$ and $\overrightarrow{r}_2$	rand() ∈ [0, 1]

Table 10 Hyperparameter tuning for RF

Parameter	Value
max-depth	NONE
min-samples-leaf	1
min-samples-split	2
max-features	'sqrt'
n-estimators	100

Table 11 Hyperparameter tuning for LightGBM

Parameter	Value	Default Value
objective	'multiclass'	binary
metric	'multi-logloss'	binary-logloss
num-class	8	1
boosting-type	'gbdt'	'gbdt'
learning-rate	0.027507197111122154	0.1
feature-fraction	0.7361822629618227	1.0
bagging-fraction	0.6911851950227005	1.0
bagging-freq	1	0
min-child-samples	61	20
lambda-l1	0.004583162943858123	0.0
lambda-l2	1.94291180153658e-07	0.0
n-estimators	600	100

Hyperparameters used

For the GWO algorithm, several hyperparameters are used to determine the best combination of subsets of features; these hyperparameters are summarized in Table 9. GWO algorithm uses an approach that utilizes the "accuracy" of the Random Forest classifiers as the fitness function. The parameters settings of Random Forest are shown in Table 10. The Table 11, provides a summary of the best hyperparameters tuning for LightGBM. The hyperparameters used for the CNN model, which constitutes the second stage of the hybrid architecture, are detailed in Table 12.

Experimental results and comparison study

In this paper, we proposed using GWO for feature selection and a hybrid method composed of the LightGBM

Table 12 Hyperparameters used for the CNN model

Layer	Parameters	Description
Conv1D	filters=64, kernel size=3, activation='relu'	1D convolution layer to extract local patterns from input sequences.
MaxPooling1D	pool size=2	Reduces spatial dimension to downsample feature maps.
Dropout	rate=0.3	Randomly drops 30% of units to prevent overfitting
Flatten	——	Flattens the output into a 1D vector for the dense layers
Dropout	rate=0.4	Additional regularization before dense layers
Dense (hidden)	units=128, activation='relu'	Fully connected layer for learning higher-level representations
Dropout	rate=0.5	Dropout before output to reduce overfitting
Dense (output)	units=8, activation ='softmax'	Final classification layer for 8 or 6 classes using softmax activation
epochs	100	Number of training epochs
batch size	64	Mini-batch size
optimizer	Adam	Optimizer used

machine learning model and the CNN model for attack classification, to improve IDS performance in the IoT environment. The combination of these three algorithms allows us to increase the performance of our model in terms of accuracy relative to the different types of attacks.

As described above, we used both the CICIoT2023 and CICIoMT2024 datasets, after pre-processing to evaluate our model, employing the same subsets — train, validation, and test — to compare our proposed model against several well-known machine learning and deep learning algorithms, namely RF [38], SVM [39], DNN [40], LSTM [41], CNN [42], and the multi-head attention model [43].

Tables 13 and 14 show detailed information regarding the various performance measures and metrics, namely accuracy, precision, and F1-score. We also calculated the False Alarm Rate (FAR), Average Accuracy, and Average DR (Detection Rate) for our model and all other classifiers. For the four models based on deep learning (DNN, LSTM, CNN, and Multi-head attention), we set the epoch to a value of 500; however, we set the epoch for our model to 100 epochs. The results obtained for the different models used in the comparative study against our proposed model were obtained using all the features detailed in Tables 4 and 5 (CICIoT2023 and CICIoMT2024 respectively).

To compare the different models, including our model, we calculated the various metrics based on the test subset using the confusion matrix (Fig. 20 and Fig. 21). Regarding the CICIoT2023 dataset, as shown in Table 13 and Fig. 16, our model achieved the highest values across all global metrics, namely an Accuracy of 95.24%, a Precision of 95.22%, an F1-Score of 95.09%, an Average Accuracy of 87.93%, and an Average Detection Rate (DR) of 86.39%.

Similarly, for the CICIoMT2024 dataset Table 14 and Fig. 17, our model also outperformed the comparative methods, achieving an Accuracy of 99.50%, a Precision of 99.52%, an F1-Score of 99.51%, an Average Accuracy of 93.22%, and an Average Detection Rate (DR) of 92.36%.

Furthermore, our model exhibited a significantly low False Alarm Rate (FAR) of 1.30% for the CICIoT2023 dataset (Fig. 16d) and 2.45% for the CICIoMT2024 dataset (Fig. 17d), thereby highlighting its reliability in minimizing false positive predictions (Figs. 18 and 19).

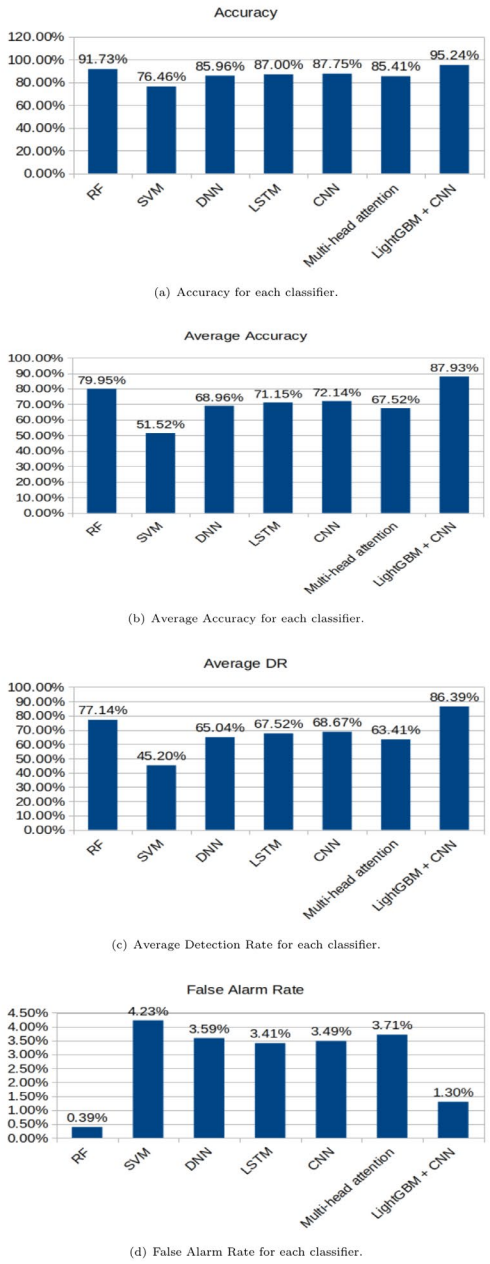
Traditional deep learning models, namely CNN, DNN, and LSTM, as well as machine learning models such as

Table 13 Performance metrics of different classifiers and our model using the CICIoT2023 dataset

Model	Accuracy	Precision	F1-Score	FAR	Average Accuracy	Average DR	Epoch
RF	91.73%	93.37%	92.28%	00.39%	79.95%	77.14%	—
SVM	76.46%	74.95%	72.82%	04.23%	51.52%	45.20%	—
DNN	85.96%	85.50%	84.98%	03.59%	68.96%	65.04%	500
LSTM	87.00%	86.78%	86.26%	03.41%	71.15%	67.52%	500
CNN	87.75%	87.34%	87.08%	03.49%	72.14%	68.67%	500
Multi-head attention	85.41%	85.33%	84.24%	03.71%	67.52%	63.41%	500
LightGBM + CNN	95.24%	95.22%	95.09%	01.30%	87.93%	86.39%	100

Table 14 Performance metrics of different classifiers and our model using the CICIoMT2024 dataset

Model	Accuracy	Precision	F1-Score	FAR	Average Accuracy	Average DR	Epoch
RF	98.44%	98.91%	98.54%	0.01%	78.68%	74.42%	—
SVM	75.21%	73.43%	70.10%	0.47%	62.49%	55.09%	—
DNN	75.59%	73.24%	73.01%	07.35%	74.69%	71.10%	500
LSTM	70.09%	72.82%	71.08%	08.57%	75.87%	72.76%	500
CNN	71.64%	72.64%	72.08%	06.59%	77.1%	73.84%	500
Multi-head attention	74.45%	72.97%	73.39%	09.14%	73.79%	70.38%	500
LightGBM + CNN	99.50%	99.52%	99.51%	2.45%	93.22%	92.36%	100



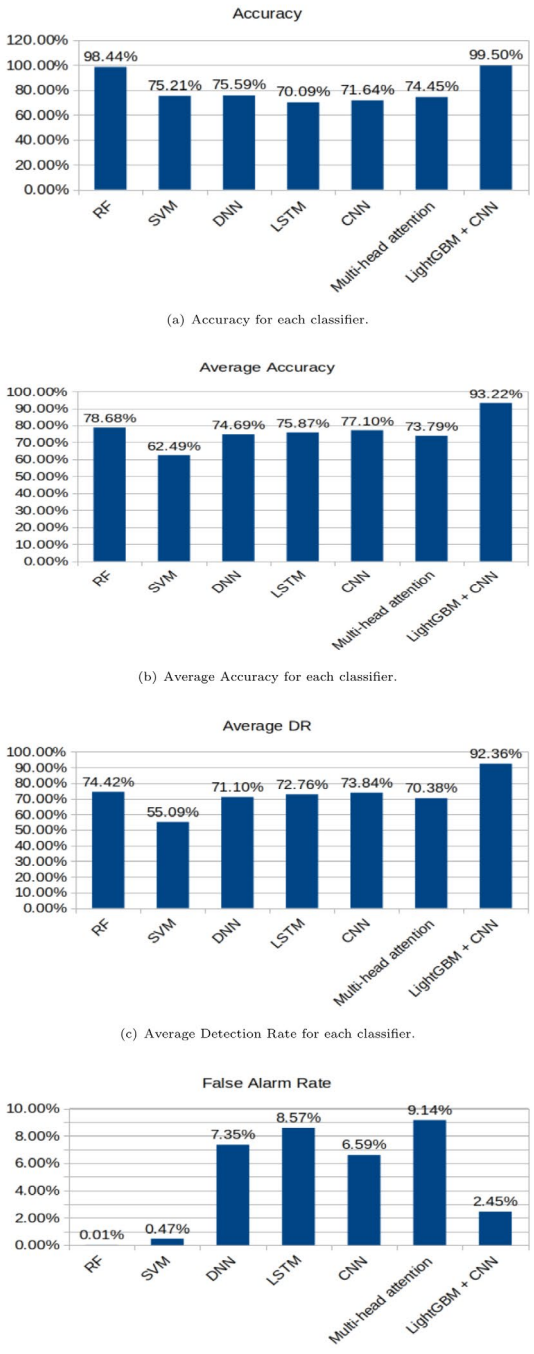
(a) Accuracy for each classifier.

(b) Average Accuracy for each classifier.

(c) Average Detection Rate for each classifier.

(d) False Alarm Rate for each classifier.

Fig. 16 Global metrics for each classifier using CICIoT2023. (a) Accuracy for each classifier. (b) Average Accuracy for each classifier. (c) Average Detection Rate for each classifier. (d) False Alarm Rate for each classifier



(a) Accuracy for each classifier.

(b) Average Accuracy for each classifier.

(c) Average Detection Rate for each classifier.

(d) False Alarm Rate for each classifier.

Fig. 17 Global metrics for each classifier using CICIoMT2024. (a) Accuracy for each classifier. (b) Average Accuracy for each classifier. (c) Average Detection Rate for each classifier. (d) False Alarm Rate for each classifier

RF and SVM, perform differently. The best-performing among them is the Random Forest (RF); however, its performance remains far below that of our hybrid model. The only metric where RF outperformed was the False Alarm Rate (FAR), achieving 0.39% and 0.01% for the CICIoT2023 dataset and the CICIoMT2024 dataset, respectively. Among the deep learning models, the best performer is the CNN model, which, for the CICIoT2023 dataset, achieved an accuracy of 87.75%, a precision of 87.34%, an F1-Score of 87.08%, a FAR of 3.49%, an average accuracy of 72.14%, and an average detection rate (DR) of 68.67% (Figs. 20 and 21).

In order to evaluate our hybrid model thoroughly, we have provided a second comprehensive comparative

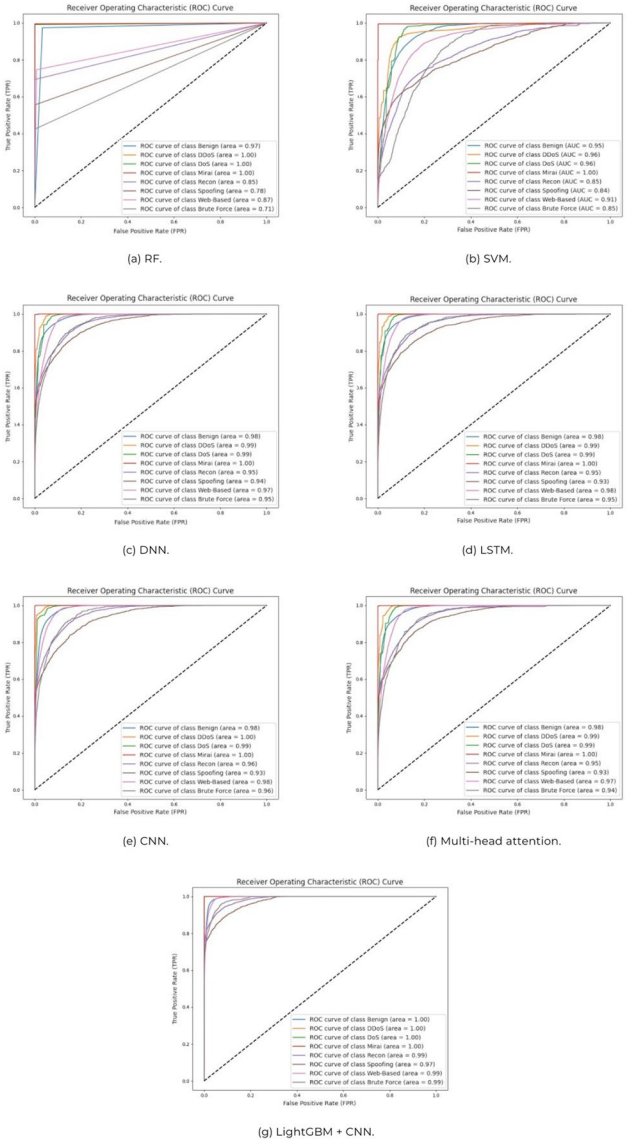


Fig. 18 ROC curves and ROC areas of different classifiers and our model using CICIoT2023. (a) ROC curves and ROC areas for classifier RF. (b) ROC curves and ROC areas for classifier SVM. (c) ROC curves and ROC areas for classifier DNN. (d) ROC curves and ROC areas for classifier LSTM. (e) ROC curves and ROC areas for classifier CNN. (f) ROC curves and ROC areas for classifiers Multi Head Attention. (g) ROC curves and ROC areas for our model LightGBM + CNN

study focused on the seven different attack categories and benign traffic from the CICIoT2023 dataset, as well as five additional attack types and benign traffic from the CICIoMT2024 dataset.

As summarized in Table 15 and Fig. 22, for the CICIoT2023 dataset, our model outperformed other methods across different classes, especially for attacks belonging to the DDoS, DoS, and Mirai categories, achieving True Positive Rates (TPR) of 99.85%, 99.87%, and 99.91%, respectively. Even for rare and less frequent attack categories, our hybrid model outperformed the other methods, particularly for attacks such as Recon, Spoofing, Web-Based, and Brute Force, with TPRs of 78.75%, 71.23%, 88.00%, and 67.18%, respectively. Only for benign traffic did the Random Forest model slightly outperform our model, achieving a TPR of 99.60% compared to our model's 98.69%. For the CICIoMT2024 dataset, as shown in Table 16 and Fig. 23, our model achieved strong results, with TPRs of 99.34%, 94.51%, 99.84%, 99.28%, and 68.86% for the MQTT, Recon, DDOS, DOS, and Spoofing, respectively.

The final metric used in this comparative study is the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC). We plotted the ROC curves, as shown in Figs. 18g (CICIoT2023) and 19g (CICIoMT2024). We calculated the AUC values to visualize the trade-off between detection and false alarm rates across different thresholds. A model is considered more effective if the ROC curves for the eight classes (Benign, DDoS, DoS, Mirai, Recon, Spoofing, Web-Based, and Brute Force) in the CICIoT2023 dataset are close to the upper left corner. For the CICIoMT2024 dataset, the ROC curves for the six classes (Benign, DDoS, DoS, Recon, Spoofing, and MQTT) are also tightly clustered near the upper left corner, indicating excellent discriminative performance across all categories.

The results show that for four classes- Benign, DDoS, DoS, and Mirai- our hybrid model achieved an AUC of approximately 1.0, indicating almost perfect classification with minimal errors. The Recon, Web-Based, and Brute Force classes also achieved very high AUC values of 0.99, reflecting near-perfect classification precision. Finally, the Spoofing class attained a strong AUC of 0.97. Additionally, our model achieves the highest average Micro and Macro ROC-AUC values, as well as the highest or near-highest ROC-AUC values for the individual classes. In contrast, for the CICIoMT2024 dataset, our model achieved an AUC close to 1.0 for all evaluated classes (Benign, DDoS, DoS, Recon, Spoofing, and MQTT), demonstrating flawless classification capability across the entire set of categories.

Overall, our model is very effective for intrusion detection. It detects and classifies various network attacks with a very high precision for the majority of classes. The performance of our model qualifies it for deployment in real Internet of Things environments for protection against cyberattacks.

Conclusion

In this paper, we proposed a hybrid approach to enhance IDS in the Internet of Things (IoT) environment, based on the meta-heuristic algorithm, the LightGBM machine learning model, and the CNN model. The combination of these three algorithms allows us to increase the performance of our model in terms of accuracy in detecting

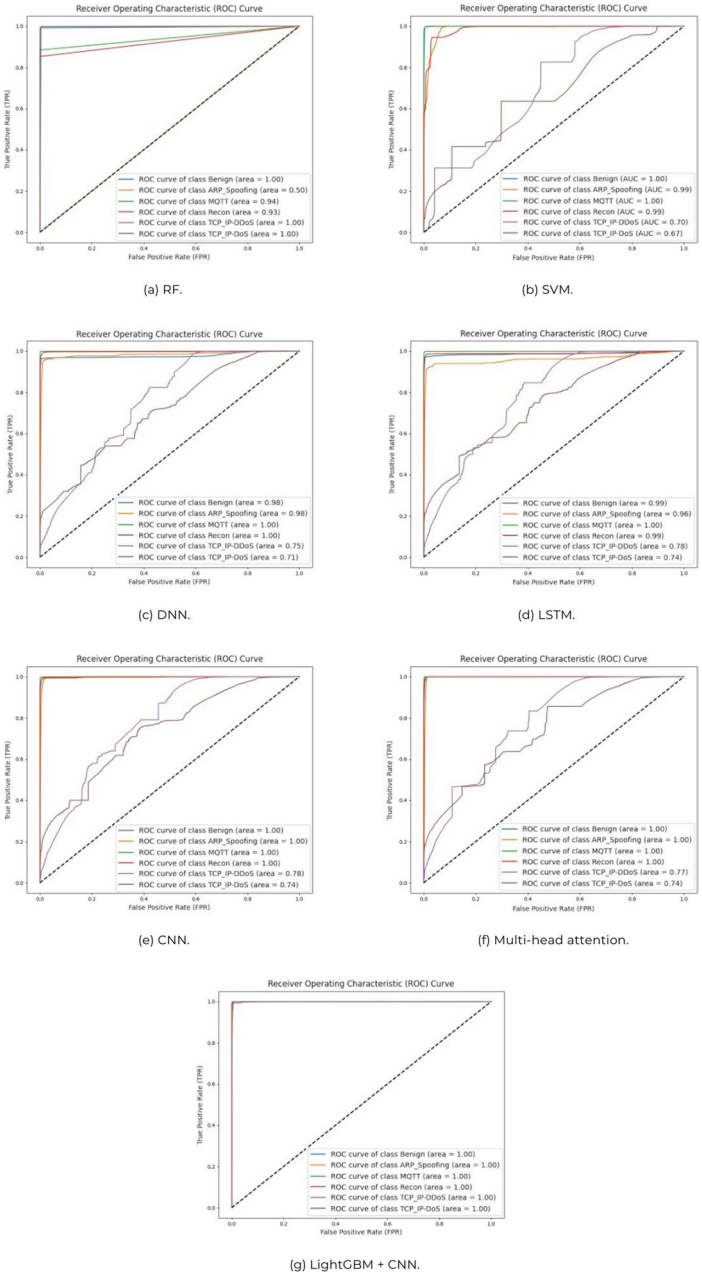


Fig. 19 ROC curves and ROC areas of different classifiers and our model using CICIoMT2024. (a) ROC curves and ROC areas for classifier RF. (b) ROC curves and ROC areas for classifier SVM. (c) ROC curves and ROC areas for classifier DNN. (d) ROC curves and ROC areas for classifier LSTM. (e) ROC curves and ROC areas for classifier CNN. (f) ROC curves and ROC areas for classifiers Multi Head Attention. (g) ROC curves and ROC areas for our model LightGBM + CNN

different types of attacks. Among several meta-heuristic algorithms, we have chosen the Grey Wolf Optimizer algorithm (GWO), which is a population-based algorithm that uses swarm intelligence to represent the function of social hierarchical leadership and the hunting process of grey wolves in their natural environment. The GWO algorithm for feature selection allows reducing the number of features by retaining those that are most essential and relevant and eliminating those that are unnecessary and redundant to improve model performance and reduce complexity and computation time.

We tested and improved our model's performance using the CICIoT2023 and CICIoMT2024 datasets, and we compared the proposed model with other deep learning algorithms, namely DNN, LSTM, CNN, and the multi-head attention transformer model. Our model achieved the highest performance on both datasets. For the CICIoT2023 dataset, it reached an accuracy of 95.24%, precision of 95.22%, F1-Score of 95.09%,

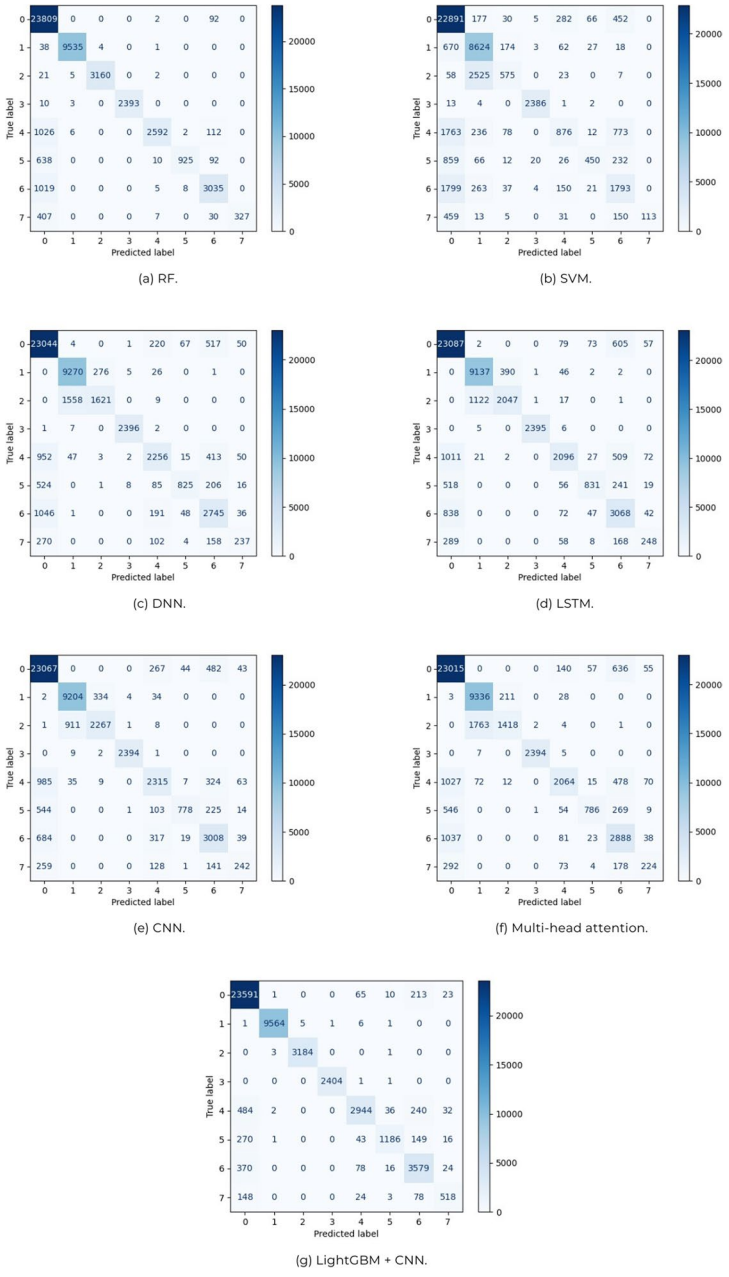


Fig. 20 Confusion Matrix of different classifiers and our model using CICIoT2023. (a) Confusion Matrix for classifier RF. (b) Confusion Matrix for classifier SVM. (c) Confusion Matrix for classifier DNN. (d) Confusion Matrix for classifier LSTM. (e) Confusion Matrix for classifier CNN. (f) Confusion Matrix for classifiers Multi Head Attention. (g) Confusion Matrix for our model LightGBM + CNN

Average Accuracy of 87.93%, and Average DR of 86.39%. For the CICIoMT2024 dataset, it achieved an Accuracy of 99.50%, a Precision of 99.52%, an F1-Score of 99.51%, an Average Accuracy of 93.22%, and an Average DR of 92.36%. Moreover, it gives a low FAR of 1.30% and 2.45% for CICIoT2023 and CICIoMT2024, respectively.

Although our model has achieved good results with a high detection rate for both the CICIoT2023 and CICIoMT2024 datasets, its performance is hindered by the insufficient representation of certain attack classes, which contain a limited number of instances in the dataset that can degrade the capacity of our model.

To mitigate the limitation arising from the underrepresentation of certain attack classes, future investigations will focus on leveraging advanced generative artificial intelligence techniques, notably Generative Adversarial Networks (GANs) and diffusion models. Such approaches are expected to enable the synthesis of realistic and diverse attack instances, thereby enhancing both the volume and heterogeneity of the dataset. A

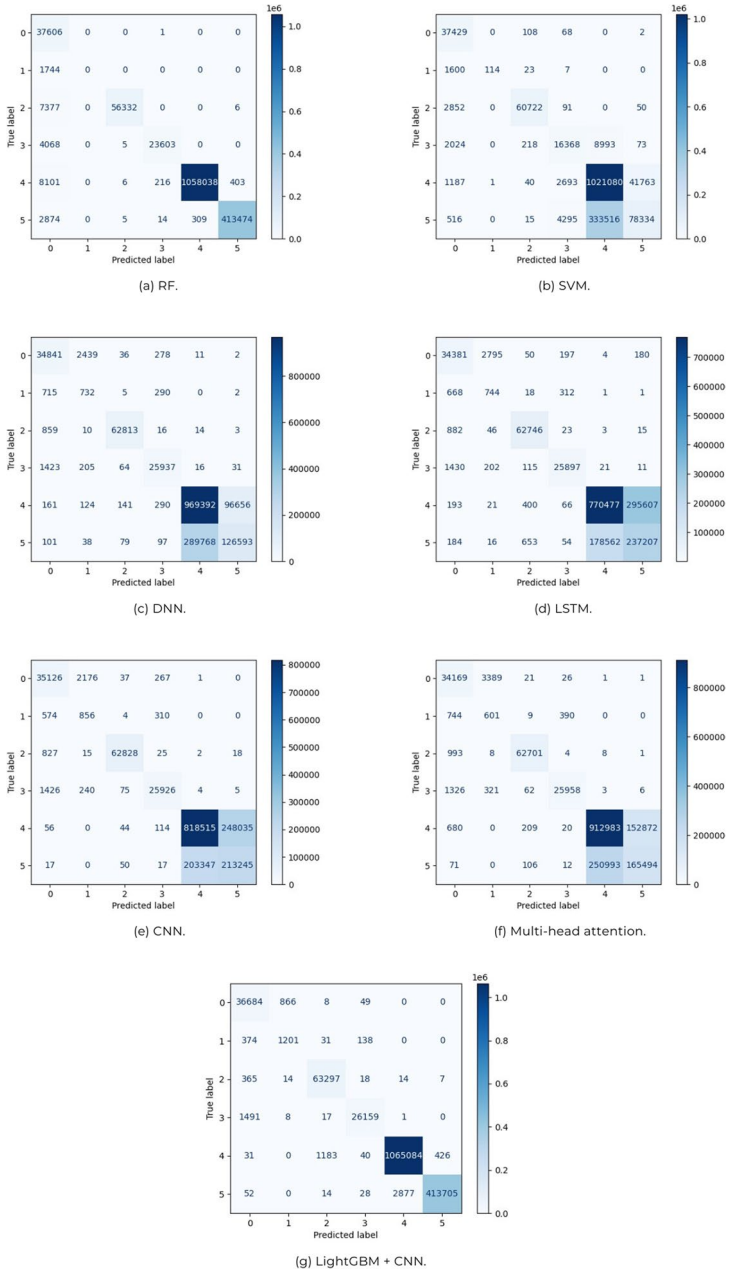


Fig. 21 Confusion Matrix of different classifiers and our model using CICIoMT2024. (a) Confusion Matrix for classifier RF. (b) Confusion Matrix for classifier SVM. (c) Confusion Matrix for classifier DNN. (d) Confusion Matrix for classifier LSTM. (e) Confusion Matrix for classifier CNN. (f) Confusion Matrix for classifiers Multi Head Attention. (g) Confusion Matrix for our model LightGBM + CNN

Table 15 True Positive Rate of each classifier and our model using the CICIoT2023 dataset

TPR	RF	SVM	DNN	LSTM	CNN	Multi-head attention	LightGBM + CNN
TPR Benign	99.60%	95.76%	96.40%	96.58%	96.5%	96.28%	98.69%
TPR DDoS	99.55%	90.03%	96.78%	95.39%	96.09%	97.47%	99.85%
TPR DoS	99.12%	18.03%	50.84%	64.20%	71.11%	44.47%	99.87%
TPR Mirai	99.45%	99.16%	99.58%	99.54%	99.50%	99.50%	99.91%
TPR Recon	69.34%	23.43%	60.35%	56.07%	61.93%	55.21%	78.75%
TPR Spoofing	55.55%	27.02%	49.54%	49.90%	46.72%	47.20%	71.23%
TPR Web-Based	74.62%	44.08%	67.49%	75.43%	73.96%	71.01%	88.00%
TPR Brute Force	42.41%	14.65%	30.73%	32.16%	31.38%	29.05%	67.18%

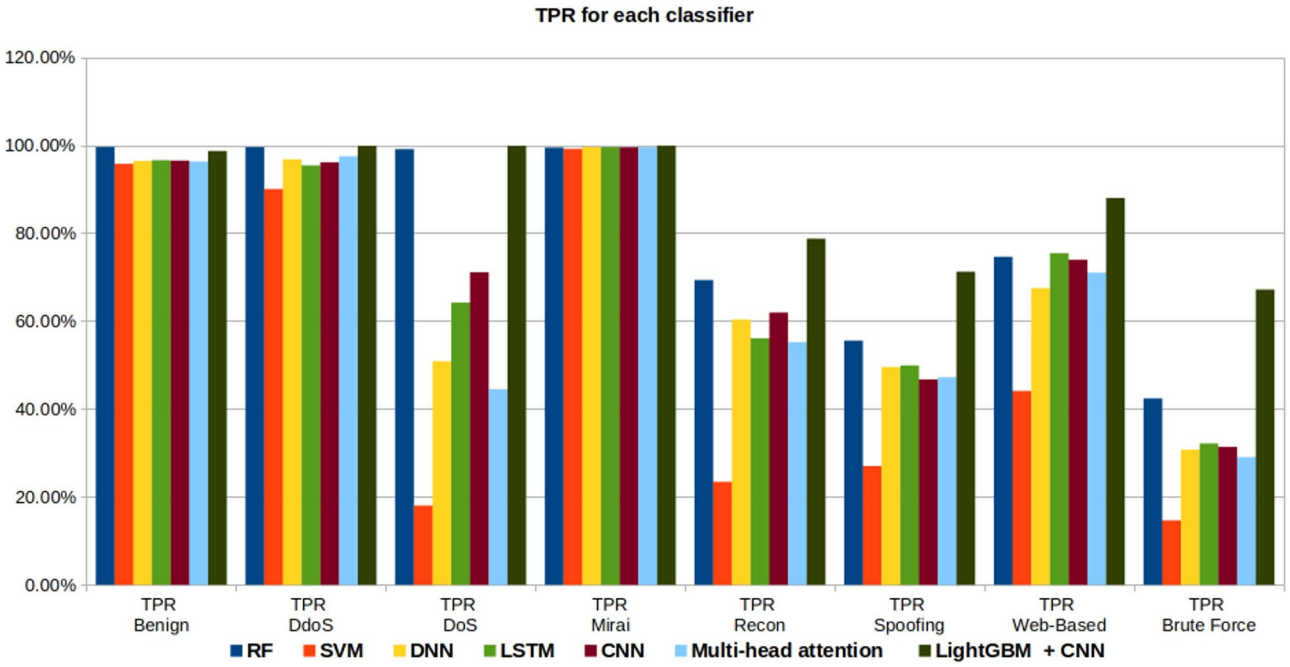


Fig. 22 True Positive Rate for each classifier using CICIoT2023

Table 16 True Positive Rate of each classifier and our model using the CICIoMT2024dataset

TPR	RF	SVM	DNN	LSTM	CNN	Multi-head attention	LightGBM + CNN
TPR Benign	99.99%	99.52%	92.64%	91.42%	93.40%	90.85%	97.54%
TPR Spoofing	0%	6.53%	41.97%	42.66%	49.08%	34.46%	68.86%
TPR MQTT	88.41%	95.30%	98.58%	98.47%	98.60%	98.40%	99.34%
TPR Recon	85.18%	59.14%	93.71%	93.57%	93.67%	93.79%	94.51%
TPR DDOS	99.18%	95.71%	90.87%	72.22%	76.72%	85.58%	99.84%
TPR DOS	99.23%	18.79%	30.38%	56.92%	51.17%	39.71%	99.28%

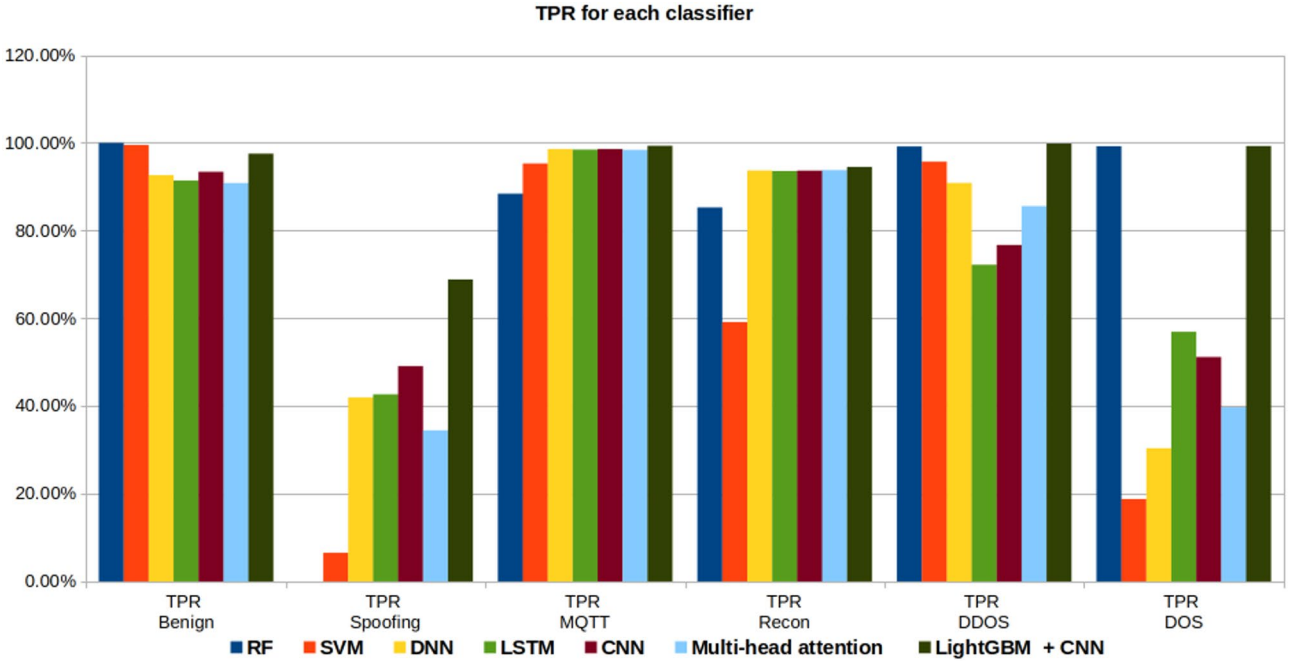


Fig. 23 True Positive Rate for each classifier using CICIoMT2024

systematic evaluation of the influence of these synthetic data on model performance, generalization capabilities, and robustness will constitute a key research direction. Furthermore, we aim to extend our approach to more heterogeneous and dynamic IoT environments, taking into account real-time computing constraints.

Authors' contributions
T.A. wrote the main manuscript text and conducted the experiments. A.A. contributed to different parts of the manuscript and experiments and guided T.A. through various stages of writing, experimentation, and result analysis. F.M. reviewed and corrected the manuscript. D.C. reviewed, corrected errors, and prepared some figures. I.U., M.A., and R.A. reviewed and corrected the manuscript.

Funding
This research received no external funding.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Competing interests
The authors declare no competing interests.

Received: 25 May 2025 / Accepted: 9 September 2025
Published online: 26 November 2025

References

1. Al-Jarrah OY, Alhussein O, Yoo PD, Muhaidat S, Taha K, Kim K (2015) Data randomization and cluster-based partitioning for botnet intrusion detection. IEEE Trans Cybern 46(8):1796–1806. https://doi.org/10.1109/TCYB.2015.2490802
2. Zhou Y, Cheng G, Jiang S, Dai M (2020) Building an efficient intrusion detection system based on feature selection and ensemble classifier. Comput Netw 174:107247. https://doi.org/10.1016/j.comnet.2020.107247
3. Liao HJ, Lin CHR, Lin YC, Tung KY (2013) Intrusion detection system: a comprehensive review. J Netw Comput Appl 36(1):16–24. https://doi.org/10.1016/j.jnca.2012.09.004
4. Ahmim A, Maazouzi F, Ahmim M, Namane S, Dhaou IB (2023) Distributed denial of service attack detection for the internet of things using hybrid deep learning model. IEEE Access 11:119862–119875. https://doi.org/10.1109/access.2023.3327620
5. Sakraoui S, Ahmim A, Derdour M, Ahmim M, Namane S, Dhaou IB (2024) Fbmp-ids: Fl-based blockchain-powered lightweight mpc-secured ids for 6g networks. IEEE Access. https://doi.org/10.1109/access.2024.3435920
6. Chebrolu S, Abraham A, Thomas JP (2005) Feature deduction and ensemble design of intrusion detection systems. Comput Secur 24(4):295–307. https://doi.org/10.1016/j.cose.2004.09.008
7. El-Hasnony IM, Barakat SI, Elhoseny M, Mostafa RR (2020) Improved feature selection model for big data analytics. IEEE Access 8:66989–67004. https://doi.org/10.1109/ACCESS.2020.2986232
8. Sp RM, Maddikunta PKR, Parimala M, Koppu S, Gadekallu TR, Chowdhary CL, Alazab M (2020) An effective feature engineering for dnn using hybrid pca-gwo for intrusion detection in iomt architecture. Comput Commun 160:139–149. https://doi.org/10.1016/j.comcom.2020.05.048
9. Narayana Rao K, Venkata Rao K, Prasad Reddy PVGD (2021) A hybrid intrusion detection system based on sparse autoencoder and deep neural network. Comput Commun 180:77–88. https://doi.org/10.1016/j.comcom.2021.08.026
10. Nazir A, Khan RA (2021) A novel combinatorial optimization based feature selection method for network intrusion detection. Comput Secur 102:102164. https://doi.org/10.1016/j.cose.2020.102164
11. Liu J, Yang D, Lian M, Li M (2021) Research on intrusion detection based on particle swarm optimization in iot. IEEE Access 9:38254–38268. https://doi.org/10.1109/ACCESS.2021.3063671
12. Keserwani PK, Govil MC, Pilli ES, Govil P (2021) A smart anomaly-based intrusion detection system for the internet of things (iot) network using gwo-pso-rf model. J Reliab Intell Environ 7(1):3–21. https://doi.org/10.1007/s40860-020-00126-x
13. Kunhare N, Tiwari R, Dhar J (2022) Intrusion detection system using hybrid classifiers with meta-heuristic algorithms for the optimization and feature selection by genetic algorithm. Comput Electr Eng 103:108383. https://doi.org/10.1016/j.compeleceng.2022.108383
14. Ewees AA, Mostafa RR, Ghoniem RM, Gaheen MA (2022) Improved seagull optimization algorithm using lévy flight and mutation operator for feature selection. Neural Comput Appl 34(10):7437–7472. https://doi.org/10.1007/s00521-021-06751-8
15. Sajith P, Nagarajan G (2022) Intrusion detection system using deep belief network & particle swarm optimization. Wirel Pers Commun 125(2):1385–1403. https://doi.org/10.1007/s11277-022-09609-x
16. Moizuddin M, Jose MV (2022) A bio-inspired hybrid deep learning model for network intrusion detection. Knowl-Based Syst 238:107894. https://doi.org/10.1016/j.knosys.2021.107894
17. Dey AK, Gupta GP, Sahu SP (2023) Hybrid meta-heuristic based feature selection mechanism for cyber-attack detection in iot-enabled networks. Procedia Comput Sci 218:318–327. https://doi.org/10.1016/j.procs.2023.01.014
18. Dey AK, Gupta GP, Sahu SP (2023) A metaheuristic-based ensemble feature selection framework for cyber threat detection in iot-enabled networks. Decis Analytics J 7:100206. https://doi.org/10.1016/j.dajour.2023.100206
19. Sanju P (2023) Enhancing intrusion detection in iot systems: a hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks. J Eng Res 11(4):356–361. https://doi.org/10.1016/j.jer.2023.100122
20. Anushiya R, Lavanya V (2023) A new deep-learning with swarm based feature selection for intelligent intrusion detection for the internet of things. Meas Sensors 26:100700. https://doi.org/10.1016/j.measen.2023.100700
21. Jayasankar T, Kiruba Buri R, Maheswaravenkatesh P (2024) Intrusion detection system using metaheuristic fireworks optimization based feature selection with deep learning on internet of things environment. J Forecast 43(2):415–428. https://doi.org/10.1002/for.3037
22. Turukmane AV, Devendiran R (2024) M-multisvm: an efficient feature selection assisted network intrusion detection system using machine learning. Comput Secur 137:103587. https://doi.org/10.1016/j.cose.2023.103587
23. Shanbhag A, Vincent S, Gowda SB, Kumar OP, Francis SAJ (2024) Leveraging metaheuristics for feature selection with machine learning classification for malicious packet detection in computer networks. IEEE Access. https://doi.org/10.1109/access.2024.3362246
24. Geetha R, Jegatheesan A, Dhanaraj RK, Vijayalakshmi K, Nayyar A, Arulkumar V, Velmurugan J, Thavasimuthu R (2024) Cvs-fln: a novel iot-ids model based on metaheuristic feature selection and neural network classification model. Multimed Tools Appl. https://doi.org/10.1007/s11042-024-19617-7
25. Saheed YK, Abdulganiyu OH, Ait Tchakoucht T (2024) Modified genetic algorithm and fine-tuned long short-term memory network for intrusion detection in the internet of things networks with edge capabilities. Appl Soft Comput 155:111434. https://doi.org/10.1016/j.asoc.2024.111434
26. Saheed YK, Chukwuere JE (2025) Cps-iiot-p2attention: explainable privacy-preserving with scaled dot-product attention in cyber physical system-industrial iot network. IEEE Access. https://doi.org/10.1109/ACCESS.2025.3566980
27. Saheed YK, Misra S (2025) Cps-iot-ppdnn: a new explainable privacy preserving dnn for resilient anomaly detection in cyber-physical systems-enabled iot networks. Chaos Solitons Fractals 191:115939. https://doi.org/10.1016/j.chaos.2024.115939
28. Saheed YK, Abdulganiyu OH, Ait Tchakoucht T (2023) A novel hybrid ensemble learning for anomaly detection in industrial sensor networks and SCADA systems for smart city infrastructures. J King Saud Univ-Comput Inf Sci 35(5):101532. https://doi.org/10.1016/j.jksuci.2023.03.010
29. Saheed Y, Misra S, Chockalingam S (2023) Autoencoder via dcnn and lstm models for intrusion detection in industrial control systems of critical infrastructures. In: 2023 IEEE/ACM 4th international workshop on engineering and cybersecurity of critical systems (encycris). IEEE, pp. 9–16. https://doi.org/10.1109/EnCyCriS59249.2023.00006
30. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
31. Muro C, Escobedo R, Spector L, Coppinger R (2011) Wolf-pack (*canis lupus*) hunting strategies emerge from simple rules in computational simulations. Behav Process 88(3):192–197. https://doi.org/10.1016/j.beproc.2011.09.006

32. Hajisalem V, Babaie S (2018) A hybrid intrusion detection system based on abc-afs algorithm for misuse and anomaly detection. Comput Netw 136:37–50. https://doi.org/10.1016/j.comnet.2018.02.028
33. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791
34. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst 30. Accessed: 20 January 2025. https://papers.nips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html
35. Neto ECP, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani AA (2023) Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. Sensors 23(13):5941. https://doi.org/10.20944/preprints202305.0443.v1
36. Dadkhah S, Neto ECP, Ferreira R, Molokwu RC, Sadeghi S, Ghorbani AA (2024) Ciciomt 2024: a benchmark dataset for multi-protocol security assessment in iomt. Internet Things 28:101351. https://doi.org/10.1016/j.iot.2024.101351
37. Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 eighth ieee international conference on data mining. IEEE, pp 413–422. https://doi.org/10.1109/ICDM.2008.17
38. Breiman L (2001) Random forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324
39. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297. https://doi.org/10.1007/BF00994018
40. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527
41. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
42. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386
43. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30. https://doi.org/10.48550/arXiv.1706.03762

Publisher's Note