BILINGUAL PUBLISHING GROUP

**Journal of Electronic & Information Systems**

https://journals.bilpubgroup.com/index.php/jeis

**ARTICLE**

# A Hybrid Artificial Intelligence Approach for Intrusion Detection in Wireless Sensor Networks

***Mortada Falah Badri, Mina Malekzadeh*** *

*Electrical and Computer Engineering Faculty, Hakim Sabzevari University, Sabzevar P.O. Box 397, Iran*

## ABSTRACT

Wireless sensor networks (WSNs) play a vital role in modern applications such as environmental monitoring, industrial automation, and smart infrastructure, where reliable data transmission, robustness, and energy efficiency are essential. However, their distributed architecture and constrained computational resources make them highly vulnerable to a wide range of security threats, including Blackhole, Grayhole, Flooding, and Scheduling attacks. These attacks can severely disrupt network functionality, degrade data integrity, and compromise the overall reliability of mission-critical operations. To address these challenges, we present an intrusion detection system (IDS) framework that leverages a diverse set of machine learning (ML) models, incorporating both boosting and non-boosting techniques, as well as deep learning (DL) architectures, including sequential and non-sequential designs. This diversity enables the framework to capture varied learning behaviors and decision boundaries. To further enhance detection accuracy and adaptability, Ant colony optimization (ACO) is employed as a metaheuristic tuning layer, refining hyperparameters to improve performance under the strict resource limitations typical of WSN environments. Each model is evaluated in both its baseline and ACO-optimized form, enabling a detailed comparative analysis that highlights the influence of optimization on intrusion detection effectiveness. Experimental results demonstrate that ACO significantly strengthens model resilience against diverse threats, offering an adaptive and efficient approach to securing modern WSN infrastructures.

***Keywords:*** Wireless Sensor Networks; Deep Learning; Machine Learning; Intrusion Detection System; Ant Colony Optimization

*CORRESPONDING AUTHOR:

Mina Malekzadeh, Electrical and Computer Engineering Faculty, Hakim Sabzevari University, Sabzevar P.O. Box 397, Iran;
Email: m.malekzadeh@hsu.ac.ir

ARTICLE INFO

Received: 25 February 2026 | Revised: 18 June 2026 | Accepted: 10 July 2026 | Published Online: 20 July 2026
DOI: https://doi.org/10.30564/jeis.v8i2.13209

CITATION

Badril, M.F., Malekzadeh, M., 2026. A Hybrid Artificial Intelligence Approach for Intrusion Detection in Wireless Sensor Networks. Journal of Electronic & Information Systems. 8(2): 43–60. DOI: https://doi.org/10.30564/jeis.v8i2.13209

# 1. Introduction

Wireless sensor networks (WSNs) are widely deployed across a range of critical applications, including real-time monitoring, data collection, and intelligent decision-making in domains such as environmental sensing, healthcare, industrial automation, and military surveillance. A typical WSN consists of spatially distributed, battery-powered sensor nodes that communicate wirelessly to gather, process, and relay information. Their decentralized architecture, scalability, and seamless integration with the Internet of Things (IoT) make WSNs a foundational infrastructure for building many adaptive and responsive systems [1]. However, the open communication channels combined with the severe computational, memory, and energy constraints of sensor nodes render WSNs highly susceptible to security threats such as Blackhole, Grayhole, Flooding, and Scheduling attacks as different forms of denial-of-service (DoS) [2]. Such attacks can rapidly drain node batteries, exhaust memory, and saturate communication bandwidth, which can effectively disable the affected nodes [3]. The consequences are especially severe in sensitive WSN environments. When WSNs are deployed for real-time monitoring in healthcare applications, attacks can block or distort data flow, leading to data loss, missed alerts, or false readings. In military scenarios, compromised networks can result in the loss of situational awareness or intelligence. Similarly, in industrial automation, where WSNs are integral to automation and control, the attacks can halt production lines, damage equipment, or even endanger human safety. Moreover, by manipulating routing protocols or injecting malicious packets, the attacks can fragment the network, isolate nodes, and undermine coordinated sensing, often without triggering conventional security alarms [4].

Thus, protecting WSNs using trust-based models and robust security frameworks against these threats is highly important. Effective protection must not only preserve constrained resources but also ensure network reliability and data integrity, thereby guaranteeing the continuity of operations [5]. To address these challenges, this work proposes an AI-based intrusion detection system (IDS) enhanced with an intelligent optimization approach to achieve an optimal balance between detection accuracy and computational efficiency, thus fitting in resource-constrained WSNs. The key contributions of this work are as follows:

- We develop a comparative intelligent IDS framework to detect and mitigate security threats in WSN environments, specifically addressing blackhole, grayhole, flooding, and scheduling attacks. The IDS incorporates 14 diverse machine learning and deep learning models, encompassing feedforward, recurrent, boosting, and non-boosting groups. This architectural diversity enables adaptive detection across heterogeneous traffic patterns and attack scenarios, thereby enhancing resilience and reliability in WSN environments.
- To further improve detection efficacy in resource-limited settings, we integrate ant colony optimization (ACO) as a metaheuristic optimization layer, applied individually to each model within the IDS.
- We present a dual-mode evaluation methodology. Every model is implemented and evaluated in both its baseline (non-optimized) and ACO-enhanced (optimized) configurations. This design enables a direct, fair, and quantifiable analysis of the optimization's impact on detection performance.
- We provide a comparative performance analysis for both intra-group and inter-group evaluation. Experiments are conducted using a multi-dimensional set of metrics to serve two primary objectives: (1) to identify the relative strengths, limitations, and operational trade-offs of each learning model within the WSN-IDS context, ultimately determining the most effective model for real-world deployment, and (2) to quantify the performance gains introduced by the optimization approach, demonstrating its consistent and statistically significant value in elevating baseline models.

Together, these contributions aim at ultimately enhancing WSN environments against the attacks that target their functionality and operational reliability. The remainder of this work is organized as follows. Section 2 reviews the related works. Section 3 describes the proposed framework. Section 4 presents the experimental results. Section 5 concludes the work.

# 2. Related Works

Securing WSNs requires addressing their unique constraints. Their dynamic topology, where nodes frequently join, leave, or move, complicates the maintenance of secure

communication and increases vulnerability to attacks. The shared wireless medium further elevates the risk of interference and data disruption. When combined with strict limitations on energy, computation, and memory, it becomes clear that no single defense method can fully safeguard these networks. Consequently, research has explored a wide range of intelligent mechanisms to protect WSNs against these challenges. In their work, Mahadik et al.[6] propose an edge heterogeneous IoT (HetIoT) defense IDS aimed at mitigating DDoS attacks by detecting and blocking malicious traffic near the network edge. The study evaluates five ML classifiers (ID3, Naive Bayes, Random Forest, Logistic Regression, and AdaBoost) alongside a hybrid DL model (CNN + LSTM). Using the CICDDoS2019 dataset, both binary and multiclass classification tasks are performed, with the hybrid model achieving superior results. However, the scope is defined by a limited set of ML algorithms and a single hybrid deep learning design. No optimization layer is applied to tune or adapt the models, leaving performance dependent on default configurations. The focus on DDoS attacks outside WSN environments further narrows the scope, leaving broader model diversity, optimization strategies, attack categories, and network contexts limited. A security mechanism to protect the ripple routing protocol (RPL) against blackhole attacks in WSN is proposed by Sharma et al.[7]. The approach leverages 6LoWPAN features and applies a distributed timer-based detection scheme. The evaluation in the Cooja simulator shows high accuracy, with reduced packet loss and true positive rates reaching 100%. However, the attack diversity remains narrow, as it is not extended to other categories of attacks beyond blackhole detection, which restricts its ability to address the wide range of threats typically encountered in WSNs. In addition, the design does not incorporate artificial intelligence (AI)-driven learning models that could enable adaptive detection or continuous improvement of classification accuracy over time.

Nordin and Mohd Pozi[8] present a survey of security threats in WSNs, focusing on how various attacks exploit vulnerabilities across different network layers. The paper discusses energy-efficient protocols and multi-channel communication strategies, then reviews common attack types such as spoofing, eavesdropping, jamming, sinkhole, wormhole, blackhole, Sybil, and DoS. A cross-layer IDS is considered a potential strategy for mitigating these threats, with attention given to the challenges of designing energy-efficient IDS solutions for resource-constrained WSNs. However, the work remains a review study and does not implement or experimentally validate protection models to quantify their effectiveness against attacks. No AI-based learning methods or optimization techniques are incorporated, and the contribution is limited to conceptual analysis rather than practical evaluation. An optimized deep neural network (DNN) model, with parameters selected using adaptive particle swarm optimization (PSO), is proposed by Ramesh et al.[9] to detect DoS attacks in wireless multimedia sensor networks (WMSNs). The results demonstrate that the proposed DNN can effectively identify DoS threats and improve network reliability under attack conditions. However, the scope of the work is limited to DoS attacks in multimedia sensor environments, without extending to other categories of threats that commonly affect WSNs, such as blackhole, grayhole, or scheduling attacks. While a DL model is employed, the study does not explore a broader range of DL algorithms or ML approaches, which restricts the generalizability of the approach across heterogeneous attack types and diverse WSN contexts, where flexibility and scalability are critical.

Using the WSN DS dataset, Liu et al.[10] suggest an intelligent IDS for WSNs to detect DoS attacks. The IDS is built exclusively on the k-nearest neighbor (KNN) algorithm, combined with three optimization strategies: particle swarm optimization (KNN PSO), arithmetic optimization (KNN AOA), and a parallel Lévy flight enhanced variant (KNN PL AOA). The results show a better performance with the optimized KNN, achieving 99% accuracy and a nearly 10% improvement over the baseline KNN classifier. By focusing solely on KNN and its three optimized variants, the study overlooks the potential benefits of ML algorithmic diversity, such as ensemble methods or decision trees, which may capture different attack patterns and improve generalization. Furthermore, the absence of deep learning models further limits the exploration of hierarchical feature extraction, which can be effective in handling complex intrusion scenarios. El Kouari et al.[11] address intrusion detection in the Industrial Internet of Things (IIoT). The work presents an industrial cybersecurity strategy built on segmented network architecture, collaboration between IT and OT teams, and vertical honeypots deployed across Industry 4.0 levels. The system integrates Wazuh for log transmission and proactive

threat response, while Snort monitors network traffic. Furthermore, Wazuh is reinforced with Elasticsearch and Kibana to provide a security information and event management (SIEM) solution, enabling data analysis, compliance enforcement, and continuous adaptation through custom rulesets and cybersecurity threat intelligence (CTI) integration. This architecture demonstrates a practical defense framework for IIoT environments. It relies on traditional IDS tools such as Snort and does not incorporate intelligent ML or DL models that could enable adaptive detection and detection accuracy.

Evaluation of using different datasets is provided by Gebreyesus et al.[12]. The UNSW-NB, WSN-DS, NSL-KDD, and CIC-IDS 2018 datasets are used as inputs to a multilayer perceptron (MLP) DL model to protect WSN environments against routing attacks, including Sybil, Wormhole, Sinkhole, and Blackhole. The use of MLP provides strong detection capabilities and demonstrates high accuracy across the four datasets. However, the reliance on a single deep learning model restricts model diversity, as alternative architectures such as convolutional or recurrent neural networks are not explored. Moreover, machine learning models are not taken into account, which limits comparative evaluation and the opportunity to balance detection accuracy with computational efficiency. Osanaiye et al.[13] propose a statistical control-based approach to detect DoS jamming attacks in WSNs. The method employs an exponentially weighted moving average (EWMA) to monitor anomalous changes in packet inter-arrival times, enabling the detection of jamming events that disrupt legitimate communication. Results from trace-driven simulations demonstrate that the proposed solution can efficiently and accurately identify jamming attacks with minimal overhead, making it lightweight for resource-constrained sensor nodes. However, the scope of the work is limited to jamming-based DoS attacks and does not extend to other categories of threats that commonly affect WSNs. Moreover, the approach relies solely on statistical techniques without incorporating intelligent approaches such as machine learning, deep learning, or optimization strategies that could enhance adaptability and scalability.

Almomani and Alenezi[14] conduct an empirical study on intrusion detection in WSNs, focusing on DoS and routing attacks using the WSN DS dataset. Seven ML techniques are examined, including Naive Bayes (NB), decision trees (DT), random forests, support vector machine (SVM), J48, k-nearest neighbor (KNN), and Bayesian Networks, along with artificial neural networks (ANN). However, while the study provides a thorough comparison of several ML models with the ANN model, it remains constrained to the selected set of models. Other deep learning architectures and broader optimization strategies are not investigated, which limits the system's adaptability to evolving attack patterns across diverse WSN scenarios. Elsadig[15] provides a comparison of the decision tree (DT) classifier with the random forest (RF), extreme gradient boosting (XGBoost), and k nearest neighbour (KNN) classifiers for the detection of DoS attacks using the WSN-DS dataset. The proposed DT-based approach is enhanced with Gini feature selection. However, the study confines its evaluation to this narrow set of ML models and does not extend the analysis to other machine learning techniques, such as ensemble hybrids, probabilistic classifiers, or advanced methods that could capture different attack behaviours. Furthermore, deep learning architectures, which can be effective in automatically extracting hierarchical features and adapting to complex intrusion scenarios, are not considered. The absence of optimization strategies also limits the adaptability of the proposed IDS to balance detection accuracy with computational efficiency.

Vinayakumar et al.[16] discuss the use of deep neural networks (DNNs) to develop an IDS capable of detecting and classifying unforeseen cyberattacks at both the network and host levels. It emphasizes the challenges posed by the dynamic nature of malware and the need for scalable solutions, highlighting the importance of systematically updating and benchmarking publicly available datasets. Evaluation is conducted using DNNs and classical machine learning classifiers across multiple benchmark datasets, including KDDCup99, NSL KDD, UNSW NB15, Kyoto, WSN DS, and CIC-IDS 2017. However, while the study confirms the effectiveness of DNNs, it focuses primarily on a single deep learning architecture without extending the analysis to other DL approaches. It does not incorporate optimization strategies to balance detection accuracy with computational efficiency. To protect WSNs against attacks from malfunctioning nodes, a routing method based on a Convolutional Neural Network coupled with fuzzy logic (CNN-FL) is presented by Shanmathi et al.[17]. The method employs a routing strategy enhanced by a neuro genetic optimizer (NGO), built on the LEACH routing protocol and utilizing a roulette wheel selection mechanism.

The authors note that the proposed method not only ensures secure data transmission but also significantly reduces latency and energy consumption. The machine learning models are used by Al Sukkar and Al-Sharaeh [18] to identify DoS attacks and mitigate their effects in WSNs. Two datasets, WSN-DS and WSN-BFSF, serve as the basis for evaluation. The results indicate that the soft ensemble technique slightly outperforms the hard ensemble technique in terms of accuracy, although other performance metrics are not considered. The security challenges in WSNs, which can lead to energy inefficiency, are discussed by Baswaraj et al. [19]. To address this, the DeepNR strategy, based on deep reinforcement learning (DRL) using a deep neural network (DNN), is proposed. Experimental results demonstrate that DeepNR outperforms traditional techniques in both security and energy efficiency. A machine learning-based framework to enhance the security of the Routing Protocol for Low-Power and Lossy Networks (RPL) is presented by Wakili et al. [20]. The approach integrates a random forest model for accurate traffic classification, a reinforcement learning module for dynamic and adaptive routing, and a modified RPL objective function that incorporates classification outcomes into routing decisions. The high detection rate and minimal false positives demonstrate improved security for the RPL protocol while enhancing QoS in IoT-WSN networks. Securing IoT with the support of WSNs is explored by Karthikeyan et al. [21]. Machine learning combined with the Firefly Algorithm (FA-ML) is employed for intrusion detection in WSN-IoT networks. The FA-ML method uses a Support Vector Machine (SVM) classifier with parameter tuning via a Grey Wolf Optimizer (GWO), evaluated using the NSL-KDD dataset. Results indicate that FA-ML outperforms Random Forest, XGBoost, and KNN.

Cluster-based routing protocols using Ant Colony Optimization and the Firefly Algorithm are proposed in EECRAIFA to maximize WSN lifetime by Wang et al. [22]. The results demonstrate effective balancing of network energy consumption, extension of network lifetime, and improved throughput. A Modified Ant Colony Optimization Algorithm (MACOA) is also introduced by Tawfeek et al. [23] to improve energy efficiency, load balancing, and optimal routing in WSNs. MACOA is compared with genetic algorithms, particle swarm optimization, artificial bee colony, deep reinforcement learning, and the energy-reliable ACO routing protocol (E-RARP) in terms of network lifetime, stabilization time, energy efficiency, load balancing, and throughput. The results show that MACOA outperforms these techniques. ACO is further discussed by Razooqi et al. [24] as a well-established heuristic algorithm for routing. The authors introduce modified ACO-based routing protocols to optimize energy consumption in WSNs. Evaluation results show that the algorithm significantly outperforms conventional protocols. An optimal routing method is proposed by Kooshari et al. [25] to reduce energy consumption in WSNs. Wireless sensor nodes are clustered using the water strider algorithm (WSA) to select cluster heads for routing. A sink node, optimized via ACO, collects packets from the cluster heads and forwards them to the base station. Results indicate that this approach effectively reduces energy consumption by minimizing the distance between cluster heads.

While the reviewed studies demonstrate valuable progress in applying intelligent techniques to WSNs security, several critical gaps remain. Many existing solutions rely on a single or a narrow set of learning models, limiting the potential benefits of model diversity and the potential advantages that could be gained from alternative approaches. In addition, the integration of optimization techniques to enhance detection capabilities and efficiency remains limited. The diversity of attacks is also restricted, mostly focusing exclusively on DoS or routing-layer attacks. To address these limitations, this work proposes a comparative intelligent IDS framework that detects four major WSN-specific threats, including blackhole, grayhole, flooding, and scheduling attacks. The IDS leverages 14 diverse machine learning and deep learning models spanning feedforward networks, recurrent architectures, boosting algorithms, and non-boosting methods. Furthermore, the ACO optimization layer is applied individually to each model, enabling direct evaluation of both baseline and optimized configurations. This breadth of models, inclusion of optimization, and coverage of multiple WSN-specific attacks can make an important advancement for efficient intrusion detection in resource-constrained WSN environments.

# 3. Methodology

The methodology presented in **Figure 1** outlines the overall design and implementation process of the proposed IDS framework for mitigating intrusions in WSN environ-

ments. To implement that, all the experimental procedures were executed on a Windows 11 system equipped with a 2.4 GHz CPU, 8 GB RAM, and 4 processing cores, using PyCharm 2024.2.4 as the development environment.

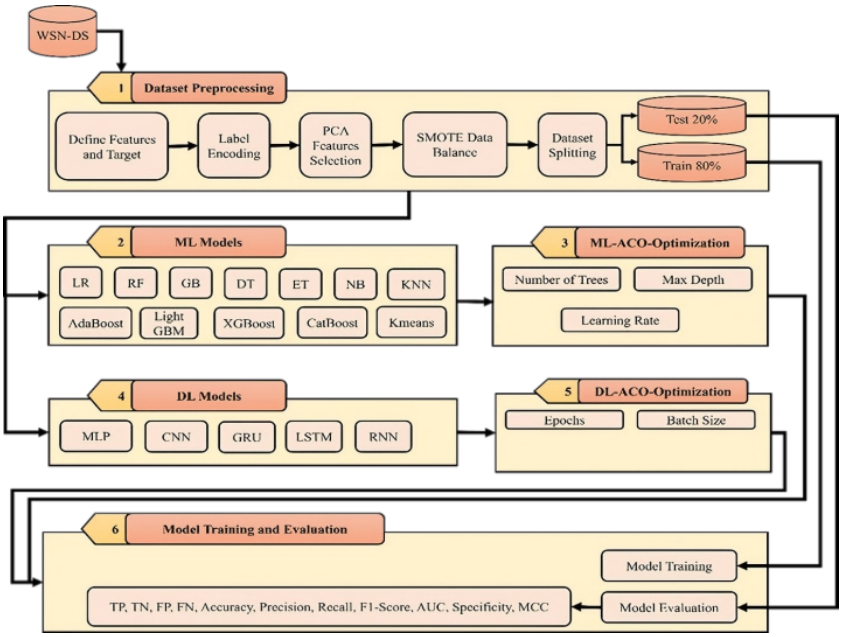


**Figure 1.** Workflow of the proposed IDS defence architecture for WSNs.

## 3.1. Dataset Preparation

The Wireless Sensor Network Dataset (WSN-DS), comprising over 374,000 records [26,27], is used for the detection and classification of four types of DoS attacks in WSN environments: blackhole, grayhole, flooding, and scheduling. The *Attack type* feature serves as the prediction target. Each record is uniquely identified by the *node ID* and *Time* features, enabling the tracking of node behavior across simulation rounds and stages. To ensure a leakage-safe preprocessing pipeline, the dataset is first divided into training and test sets using an 80/20 split. All subsequent transformations are fit exclusively on the training data and then applied to the validation and test sets using the learned parameters. Categorical features (e.g., protocol type, node ID) are encoded using LabelEncoder fit on the training set. Numerical features are standardized using StandardScaler, also fit only on the training portion. Standardization prevents continuous features with large numeric ranges, such as *Dist_To_CH*, *dist_CH_To_BS*, *Data_Sent_To_BS*, *Expanded Energy*, and *Rank*, from overshadowing subtler binary or categorical indicators such as *Is_CH*, *who CH*, *send_code*, and the ADV/JOIN/SCH/DATA send/receive flags. To reduce dimensionality and highlight the most informative structures, principal component analysis is applied with ten components (*PCA(n_components = 10)*), with the PCA model trained solely on the training data and then applied to the test set. This step filters noise and redundancy while retaining the key components that capture attack behavior. Because attack traffic is imbalanced, oversampling is performed using SMOTE only on the training portion. SMOTE is not applied to validation or test data, ensuring that synthetic samples do not leak information into evaluation data.

Several preprocessing decisions require careful justification to ensure that they do not introduce unintended biases. The *node ID* feature, while potentially sensitive to shortcut learning if treated as an ordinal variable, also captures node-specific behavioral patterns relevant to WSN operation. We therefore retain the node ID but mitigate ordering effects by applying LabelEncoder consistently across training folds and ensuring that the model cannot rely solely on identity-based shortcuts. Feature scaling is applied to prevent large-range continuous attributes from dominating the learning process, while PCA with ten components is included to reduce redundancy and highlight the most informative structure in the data. Although tree-based models are generally robust to unscaled features and may not require PCA, a unified preprocessing pipeline is applied across all model families for

consistency and comparability. SMOTE is used to address class imbalance in the training data, ensuring that minority attack types are adequately represented without affecting the validation or test sets. All steps follow established IDS practice and are implemented in a leakage-safe manner to preserve the integrity of model evaluation.

## 3.2. Model Selection

DoS attacks occur in multiple forms, ranging from simple flooding attempts to more sophisticated, distributed, and stealthy variations. Due to this diversity, a single learning-based detection approach is often insufficient to capture all attack patterns effectively. To address this challenge, our proposed IDS integrates a diverse ensemble of machine learning and deep learning models, categorized according to their structural and algorithmic characteristics, to leverage their complementary strengths. Within the domain of machine learning, we divide our models into boosting and non-boosting groups. The boosting category encompasses adaptive boosting (AdaBoost) as well as four gradient-based ensemble methods, including gradient boosting (GB), light gradient boosting machine (LightGBM), extreme gradient boosting (XGBoost), and categorical boosting (CatBoost), which are particularly effective in enhancing accuracy and efficiency. AdaBoost combines weak learners into a strong classifier by focusing on misclassified instances, which is valuable in WSNs for distinguishing subtle DoS patterns when attack signatures are not well defined. GB builds models sequentially to minimize error, capturing complex feature interactions. Its robustness to noise and flexibility make it suitable for detecting diverse behaviors in WSN traffic. LightGBM is used for speed and memory efficiency, making it well-suited for large-scale and imbalanced data. XGBoost extends GB with regularization and parallel processing, providing high accuracy and scalability for real-time detection. CatBoost, with its native handling of categorical features, reduces overfitting and generalizes well where categorical inputs such as protocol types or node IDs are common.

Non-boosting algorithms complement this by offering lightweight, interpretable solutions. Our non-boosting selection includes algorithms such as logistic regression (LR), support vector machine (SVM), k-nearest neighbors (KNN), naive bayes (NB), the k-means clustering algorithm, as well as tree-based models, including decision tree (DT), random forest (RF), and extra trees (ET). The LR provides a simple yet effective baseline for binary attack detection in resource-constrained environments. SVM performs well in high-dimensional spaces and is effective for distinguishing normal and attack traffic in smaller datasets. KNN classifies based on proximity in feature space, detecting anomalies by comparing new traffic to known patterns. NB provides a probabilistic approach that is lightweight and fast, ideal for constrained sensor nodes. The k-means is an unsupervised algorithm that can cluster similar traffic patterns and flag outliers that may indicate attack activity, even without labelled data. DTs are interpretable and fast, making them suitable for real-time WSN environments. RF is an ensemble of decision trees that improves robustness and reduces overfitting. It performs well in noisy data and can handle feature interactions critical for accurate intrusion detection. ET is similar to RF but introduces more randomness during tree construction, enhancing generalization. They are efficient and effective for detecting diverse attack patterns in WSNs with minimal tuning.

Deep learning models extend detection capabilities further by capturing complex temporal and spatial attack patterns. On the deep learning side, we differentiate between non-sequential (feedforward) and sequential (recurrent) architectures. For non-sequential modeling, we utilize the multilayer perceptron (MLP) and the convolutional neural network (CNN), both of which process input data in a single forward pass without accounting for temporal dependencies. CNNs automatically extract spatial features from traffic data, identifying abnormal packet structures or frequency distributions that signal attacks, while the MLP model nonlinear relationships in tabular features such as packet size, frequency, and delay, making them versatile for general-purpose DoS detection. In contrast, for tasks involving sequential or time-series data to capture temporal dependencies in WSN traffic flows, we adopt recurrent architectures, including the recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU), to capture temporal dynamics and long-range dependencies. RNN is effective for short-term patterns, while LSTM overcomes vanishing gradient issues to learn long-term dependencies, making them suitable for detecting stealthy or delayed DoS attacks. GRU provides similar capabilities with fewer parameters, offering computational efficiency for resource-constrained environ-

ments. Collectively, these models form an IDS framework to balance interpretability, computational overhead, and detection accuracy in WSN environments.

All DL models operate on the same PCA-transformed feature vectors used by the ML models. After preprocessing, each sample is represented as a 10-dimensional vector produced by PCA (n_components = 10). For CNN and recurrent architectures, this vector is reshaped into a fixed-length sequence of 10 time steps with a single feature channel, yielding an input shape of (10, 1). This reshaping does not impose temporal semantics; instead, it enables convolutional and recurrent layers to learn local dependencies and structured interactions among PCA components. To avoid temporal or entity leakage, sequences are constructed independently for each sample, without grouping by node ID or by time. No sequence spans multiple records, and no ordering information from the original dataset is used. This ensures that the DL models do not inadvertently access future information or mix entities across training and testing splits. The CNN architecture consists of a 1D convolutional layer with 32 filters (kernel size = 3, ReLU activation), followed by max pooling, a flatten layer, and two fully connected layers with 64 and 32 units (ReLU), including dropout (0.3) for regularization. The recurrent models (RNN, LSTM, GRU) use a single recurrent layer with 32 units, followed by a dense layer with 32 ReLU units. All deep learning models end with a softmax output layer for five-class classification. Training is performed using the Adam optimizer (learning rate = 0.001), categorical cross-entropy loss, batch size 64, and up to 50 epochs with early stopping (patience = 5). Class weighting is applied to mitigate class imbalance. The approximate parameter counts for the CNN, LSTM, and GRU models are 12,000, 18,000, and 17,000 parameters, respectively, ensuring suitability for resource-constrained WSN environments.

## 3.3. ACO Optimization

To enhance the performance of the models in the IDS framework, we employ Ant Colony Optimization (ACO) as a hyperparameter search procedure. In our implementation, ACO operates in four stages: solution construction, fitness evaluation, pheromone update, and termination, At the beginning of each iteration, every ant constructs a complete hyperparameter vector for the target model. For hyperparameter *i*, let $\Omega_i$ denote the set of candidate options. The probability that an ant selects option $j \in \Omega_i$ is given in Equation (1):

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{k \in \Omega_1} \tau_{ik}^{\alpha} \eta_{ik}^{\beta}} \tag{1}$$

where $\tau_{ij}$ is the pheromone level, $\eta_{ij}$ is the heuristic desirability, and $\alpha$, $\beta$ control their influence. By sampling one option for each hyperparameter, the ant forms a full configuration $h = (h_1, h_2, \ldots, h_m)$.

This configuration is then evaluated in the second stage. Each constructed configuration *h* is used to train the corresponding model on the training set. Performance is measured on a held-out validation split using the F1-score, which serves as the fitness value, as shown in Equation (2):

$$f(h) = \text{F1} - \text{score}_{\text{validation}} \tag{2}$$

This ensures that the optimization process directly targets generalization performance. After all ants complete their evaluations, pheromone levels are updated in two steps, evaporation and reinforcement by elite ants (top 10%), as shown in Equations (3)–(4), respectively.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \tag{3}$$

$$\begin{aligned} \Delta\tau_{ij}^{(e)} &= \begin{cases} \frac{f(e)}{1 + rank(e)}, & \text{if option } j \text{ is used in } e \\ 0, \end{cases} \\ \tau_{ij} &\leftarrow \tau_{ij} + \sum\nolimits_{e \in E} \Delta\tau_{ij}^{(e)} \qquad \text{otherwise} \end{aligned} \tag{4}$$

This reinforcement biases future ants toward high-quality hyperparameter choices. In the final stage, the optimization stops when either the maximum number of iterations is reached, or no improvement in the best fitness is observed for five consecutive iterations. The best configuration found during the search is returned as the final tuned hyperparameter set. **Algorithm 1** summarizes these four stages, followed by the ACO candidate options and constructed solutions applied to the boosting, non-boosting, and DL models within the IDS framework, as presented in **Tables 1–3**.

**Algorithm 1** ACO-Based Hyperparameter Optimization

```
procedure ACO_OPTIMIZE(model_name, search_space):
    initialize pheromone τ_ij = τ0 for all hyperparameter options j
    initialize heuristic η_ij = η0 for all options
    best_solution = None
    best_fitness = −inf
    for iteration = 1 to num_iterations do:
        solutions = []
        for ant = 1 to num_ants do:
            config = {}
            for each hyperparameter i in search_space do:
                compute probabilities P_ij from τ_ij and η_ij
                select option j according to P_ij
                config[i] = option j
            fitness = TrainAndValidate(model_name, config)
            solutions.append((config, fitness))
            if fitness > best_fitness:
                best_fitness = fitness
                best_solution = config
        elite = top E solutions by fitness
        for all i, j:
            τ_ij = (1 − ρ) × τ_ij
        for each elite solution e:
            for each hyperparameter i and chosen option j in e.config:
                τ_ij = τ_ij + Δτ_ij(e.fitness)
    return best_solution, best_fitness
end procedure
```

**Table 1.** ACO for Non-Boosting ML models.

| Model Name | ACO Candidate Options and Constructed Solutions | Best Hyperparameters (ACO Optimized) |
|---|---|---|
| **LR** | C (0.001–100), max_iter (100–1,000) | C = 18.2, max_iter = 900 |
| **SVM** | C (0.01–100), gamma ($1 \times 10^{-5}$–1), kernel {linear, rbf, poly} | C = 14.7, gamma = 0.0021, kernel = rbf |
| **KNN** | n_neighbors (1–50), weights {uniform, distance} | n_neighbors = 9, weights = distance |
| **NB** | var_smoothing ($1 \times 10^{-9}$–$1 \times 10^{-3}$) | var_smoothing = $1 \times 10^{-7}$ |
| **K-Means** | n_clusters (2–50), init {k-means++, random}, max_iter (100–1,000) | n_clusters = 8, init = k means++, max_iter = 600 |
| **DT** | max_depth (1–50), min_samples_split (2–20) | max_depth = 22, min_samples_split = 4 |
| **RF** | n_estimators (50–1,000), max_depth (3–50) | n_estimators = 460, max_depth = 32 |
| **ET** | n_estimators (50–1,000), max_depth (3–50) | n_estimators = 520, max_depth = 28 |

**Table 2.** ACO for Boosting ML models.

| Model Name | ACO Candidate Options and Constructed Solutions | Best Hyperparameters (ACO Optimized) |
|---|---|---|
| **AdaBoost** | n_estimators (50–1,000), learning_rate (0.01–2.0) | n_estimators = 520, learning_rate = 0.78 |
| **GB** | n_estimators (50–1,000), learning_rate (0.01–0.5), max_depth (3–15) | n_estimators = 680, learning_rate = 0.06, max_depth = 7 |
| **LightGBM** | n_estimators (50–2,000), learning_rate (0.001–0.5), max_depth (−1–50) | n_estimators = 1,350, learning_rate = 0.028, max_depth = 21 |
| **XGBoost** | n_estimators (50–2,000), learning_rate (0.001–0.5), max_depth (3–15) | n_estimators = 910, learning_rate = 0.045, max_depth = 9 |
| **CatBoost** | iterations (100–2,000), learning_rate (0.001–0.5), depth (3–12) | iterations = 1,200, learning_rate = 0.041, depth = 8 |

**Table 3.** ACO for DL models.

| Model Name | ACO Candidate Options and Constructed Solutions | Best Hyperparameters (ACO Optimized) |
|---|---|---|
| **CNN** | num_filters (16–512), kernel_size (2–7), num_layers (1–10), learning_rate (0.0001–0.1), batch_size (16–256) | num_filters = 128, kernel_size = 3, num_layers = 4, learning_rate = 0.0012, batch_size = 64 |
| **MLP** | num_hidden_layers (1–10), units_per_layer (16–1,024), learning_rate (0.0001–0.1), batch_size (16–256) | num_hidden_layers = 4, units_per_layer = 256, learning_rate = 0.0018, batch_size = 64 |

**Table 3.** *Cont.*

| Model Name | ACO Candidate Options and Constructed Solutions | Best Hyperparameters (ACO Optimized) |
|---|---|---|
| **RNN** | hidden_units (32–512), num_layers (1–5), learning_rate (0.0001–0.05), sequence_length (10–200) | hidden_units = 256, num_layers = 3, learning_rate = 0.0025, sequence_length = 80 |
| **LSTM** | hidden_units (32–512), num_layers (1–5), dropout (0.0–0.5), learning_rate (0.0001–0.05) | hidden_units = 256, num_layers = 3, dropout = 0.28, learning_rate = 0.0011 |
| **GRU** | hidden_units (32–512), num_layers (1–5), dropout (0.0–0.5), learning_rate (0.0001–0.05) | hidden_units = 224, num_layers = 3, dropout = 0.25, learning_rate = 0.0013 |

### 3.4. Model Training and Evaluation

Experiments are conducted to assess the effectiveness of the proposed IDS framework. The intrusion detection task is formulated as a multiclass classification problem with five classes: normal, blackhole, grayhole, flooding, and scheduling. Because true detection (TP, TN) and false detection (FP, FN) are not uniquely defined in multiclass settings, we adopt the standard one-vs-rest formulation when computing per-class precision, recall, and F1-score. It is important to note that the one-vs-rest formulation is used exclusively for metric computation; all models are trained directly on the original multiclass labels without decomposing the task into binary sub-models. The reported overall F1-score corresponds to the macro-F1 averaging method. This choice is motivated by the inherent class imbalance in the WSN intrusion-detection dataset, where certain attack types (e.g., scheduling or grayhole) appear less frequently than others. Macro-F1 assigns equal weight to each class by averaging the per-class F1-scores without considering class support, ensuring that minority classes contribute proportionally to the final metric. This provides a balanced and security-relevant assessment of detection performance across all attack categories, which is essential for evaluating IDS robustness in heterogeneous WSN environments. In addition to per-class metrics, we evaluate F1-scores to provide a balanced view of overall performance under class imbalance and accuracy to capture complementary aspects of detection capability. To further illustrate model behavior, confusion matrices are generated before and after ACO optimization. These matrices highlight which attack types are reliably detected and which remain challenging, offering deeper insight into the strengths and limitations of each model.

The experiments contribute in two distinct ways:

- Model comparison: By measuring the metrics for each model in the IDS, the comparative analysis identifies the relative strengths and weaknesses of different learning models, guiding the selection of suitable IDS approaches for real-world deployment.
- Optimization benefits: The same metrics are used to quantify the improvements achieved through ACO-based hyperparameter tuning by comparing performance before and after optimization. The measurable gains in detection accuracy and robustness underscore the value of metaheuristic strategies in resource-constrained networks.

This dual evaluation ensures that the analysis not only benchmarks individual models but also demonstrates the tangible advantages of incorporating optimization into the IDS framework.

## 4. Results and Discussion

This section presents the implementation outcomes of the proposed IDS framework across its diverse set of learning models, evaluated both before and after applying the optimization technique. To facilitate meaningful intra-group and inter-group comparisons, the models are categorized into four architectural classes: boosting-based, non-boosting, sequential, and non-sequential. Each model's behaviour is assessed based on its ability to differentiate between normal and intrusive traffic within a WSN environment. The results are structured to highlight the influence of the ACO approach on each model's predictive performance. Improvements are analyzed across multiple metrics, offering a comparative evaluation of how the IDS enhances overall decision-making quality.

### 4.1. Confusion Matrix Analysis

In intrusion detection for WSNs, true positives (TP) represent the number of actual attacks that the IDS correctly identifies. A high TP count means the IDS is effectively recognizing threats, which is essential for maintaining network

integrity. In contrast, a low TP denotes that many intrusions go undetected, leaving the system exposed and vulnerable. true negatives (TN), on the other hand, refer to the number of correctly classified benign activities. In WSNs, a high TN reflects the system's ability to preserve normal operations without triggering unnecessary alerts, which helps conserve energy and maintain trust. A low TN shows that legitimate traffic is being misclassified, potentially causing disruptions and resource waste. False positives (FP) occur when a normal packet is mistakenly flagged as malicious. In WSN environments, a high FP rate can trigger excessive alerts, leading to unnecessary energy consumption and reduced node lifespan. A low FP rate, on the other hand, indicates that the IDS avoids mislabeling safe traffic, which is essential for operational efficiency. False negatives (FN) represent actual attacks that the IDS fails to detect. In WSNs, high FN values are particularly dangerous, as undetected threats can compromise data, disrupt communication, or disable nodes. A low FN count reflects an IDS that is responsive and capable of identifying intrusions before they cause substantial harm. **Figure 2** illustrates the comparative results across models, both before and after optimization.

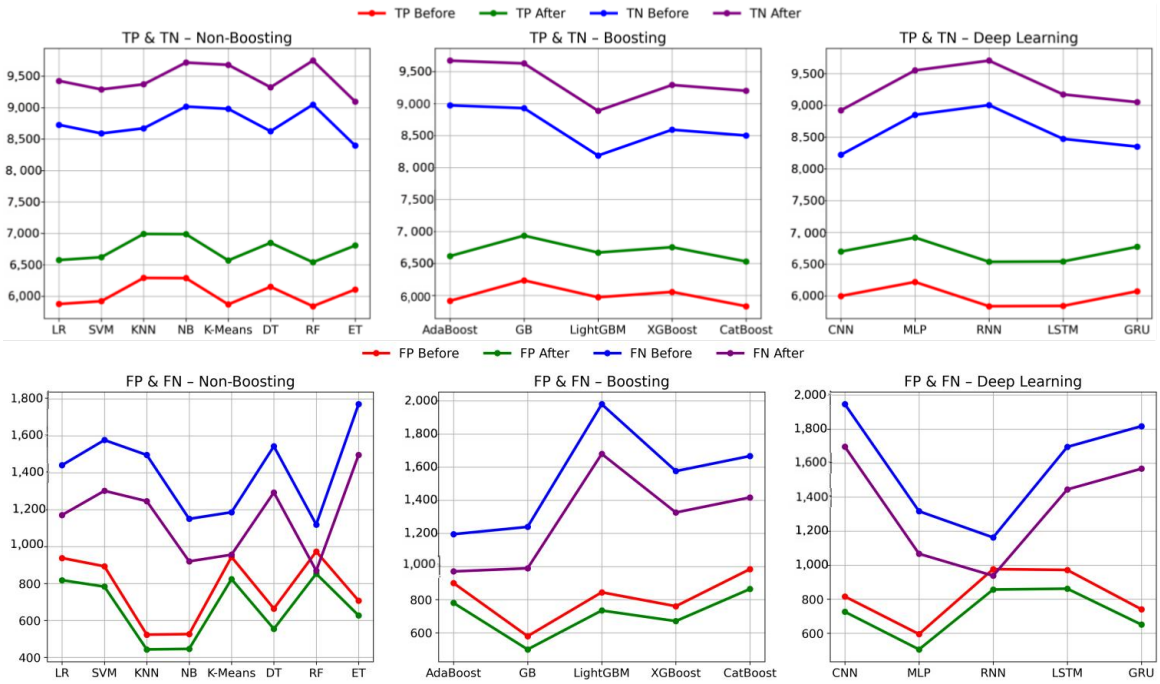


**Figure 2.** True and false detection rate of IDS pre- and post-optimization.

Among the non-boosting machine learning models, KNN and NB achieved the strongest detection performance before optimization and improved further afterward. Their effectiveness can be attributed to KNN's sensitivity to local data structures and NB's probabilistic modeling of feature distributions. NB remained one of the most balanced performers after optimization, while RF closely followed due to its ensemble-based robustness. ET initially exhibited the weakest performance, likely because its randomized split strategy limited its ability to capture subtle attack patterns; although it showed noticeable improvement after optimization. LR and SVM started with comparatively lower performance but benefited considerably from ACO, indicating that simpler models respond well to parameter and feature refinement. Within the boosting-based models, GB consistently delivered the best overall performance before and after optimization, reflecting its strength in capturing complex relationships through iterative learning. LightGBM showed the greatest improvement, overcoming weaker initial results and achieving strong post-optimization performance. XGBoost and CatBoost demonstrated balanced behavior with steady gains following optimization, while AdaBoost also improved consistently across all evaluation measures. Overall, all boosting models benefited from ACO, with GB maintaining the strongest detection capability. For deep learning models, MLP emerged as the top performer after optimiza-

tion, highlighting its ability to model complex nonlinear relationships in structured network traffic data. GRU followed closely, benefiting from its efficient handling of temporal dependencies. CNN and RNN also achieved strong results after optimization, demonstrating effectiveness in learning spatial and sequential patterns. LSTM showed comparatively smaller gains, possibly due to its architectural complexity and sensitivity to hyperparameter settings. Despite these differences, all deep learning models exhibited consistent performance improvements following optimization. It is also revealed that misclassifications are concentrated among attack types with inherently similar behavioural patterns. The most frequently confused pairs are Grayhole vs. Blackhole and Flooding vs. Scheduling. Grayhole and Blackhole both involve packet-dropping behaviour, differing mainly in selectivity, which leads to overlapping traffic signatures. Similarly, Flooding and Scheduling attacks both disrupt network load and timing characteristics, producing comparable anomalies in traffic volume and temporal patterns.

## 4.2. Accuracy

Accuracy measures the overall proportion of correct classifications, including both attacks and normal traffic. In WSNs, high accuracy indicates that the IDS is generally reliable, whereas low accuracy reflects frequent misclassifications that undermine its credibility. **Figure 3** illustrates the comparison of accuracy values before and after optimization, highlighting the improvement achieved through the approach.

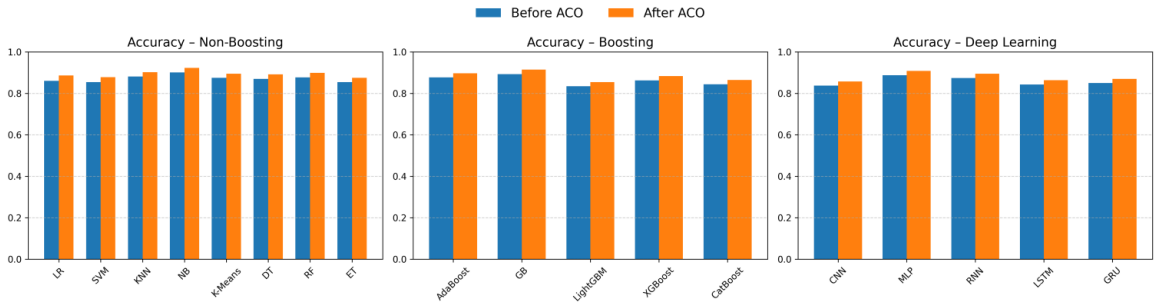


**Figure 3.** Accuracy comparison of the IDS models.

Accuracy gains were evident across all non-boosting ML models, though the magnitude varied. NB stood out with the highest accuracy before (0.9013) and after optimization (0.9226), reflecting its strength in probabilistic classification when feature distributions are well-separated. KNN also performed well, improving from 0.8811 to 0.9016, due to its sensitivity to local data structures. RF and DT showed solid post-ACO accuracy (0.899 and 0.8912), benefiting from ensemble stability and tree-based partitioning. LR and SVM, while simpler in design, still achieved respectable improvements, reaching 0.8862 and 0.8782, respectively. ET, though improved to 0.875, remained the lowest in this group, likely due to its randomized splits reducing consistency. The boosting classifiers showed noticeable improvements in accuracy after ACO optimization. GB, which started at 0.8929, climbed to 0.9139, confirming its ability to generalize well across diverse intrusion patterns. AdaBoost and XGB followed closely, reaching 0.8966 and 0.8829, respectively, both benefiting from ACO's fine-tuning of their ensemble weights. CatBoost, while initially lower at 0.8438, improved to 0.8642, suggesting that its categorical handling gained precision post-optimization. LightGBM, despite its aggressive learning strategy, remained the least accurate in this group (0.8337–0.854), likely due to its higher FN count. Overall, GB retained its lead, while others narrowed the gap through optimization. Among DL models, MLP emerged as the most accurate, rising from 0.8874 to 0.9078, a testament to its dense architecture's ability to learn complex feature interactions. RNN and CNN followed by final accuracies of 0.8944 and 0.8573, showing that sequential and spatial modeling both benefit from ACO's parameter refinement. GRU and LSTM, while improved (0.8698 and 0.8631), lagged slightly behind, possibly due to their deeper architectures requiring more extensive tuning. Despite architectural differences, all DL models showed meaningful gains, with MLP leading the pack in overall classification reliability.

## 4.3. Precision

Precision reflects how many of the predicted intrusions are actually true intrusions. In WSNs, high precision means the IDS generates correct alerts, minimizing false alarms and conserving energy. Low precision implies that many alerts are incorrect, which can lead to wasted resources and reduced confidence in the system's decisions. **Figure 4** illustrates the precision outcomes across models, highlighting gains in alert reliability.

Precision results among the non-boosting ML models showed NB and KNN emerged as the most precise, both reaching 0.94 after optimization. This is due to NB's probabilistic clarity and KNN's local sensitivity, which together help avoid false alarms. DT and ET followed closely, with final scores of 0.9252 and 0.9157, strong outcomes for tree-based learners that often struggle with overfitting. RF, despite its ensemble strength, achieved 0.8847, suggesting that its broader variance may have introduced more borderline decisions. Meanwhile, LR and SVM, though simpler in form, improved to 0.8894 and 0.8943. These results show that the IDS models can achieve competitive precision when tuned effectively with ACO. In the boosting group, GB proved its consistency with the highest results while also improving its precision from 0.9150 to 0.933. XGBoost and LightGBM followed, both crossing the 0.90 threshold post-optimization, an indicator that ACO helped refine their thresholding and reduce misclassifications. AdaBoost and CatBoost, though slightly more conservative in their gains, still showed good improvements, with final precision values of 0.8945 and 0.8832. Thus, while all boosting models benefited, GB's balance of depth and regularization gave it a slight edge in precision-focused performance. Within the deep learning group, MLP achieved the highest precision, reaching 0.9319 after ACO optimization. Its dense, fully connected layers excelled at isolating relevant features and reducing false alarms. GRU and CNN also performed strongly, with post-ACO precision values of 0.9123 and 0.9023, respectively, leveraging GRU's temporal efficiency and CNN's spatial acuity. RNN and LSTM, while improved, settled just below the 0.89 value. Their slightly lower precision can be attributed to their sensitivity to sequence noise or longer convergence times. Collectively, these results show that ACO not only improved precision across models but also revealed their potential to make more reliable decisions. In the context of WSN intrusion detection, this translates into fewer false alarms and greater trust in system alerts.

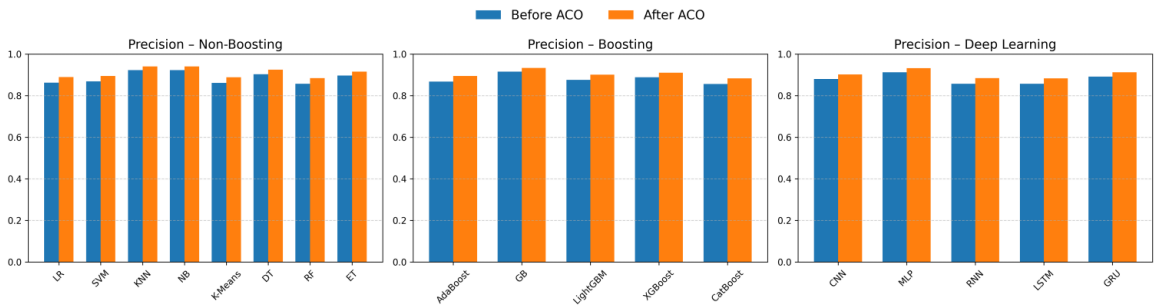


**Figure 4.** Precision comparison.

## 4.4. Recall

Recall indicates how many actual intrusions were successfully detected. In WSNs, high recall ensures that most threats are intercepted. Low recall means the IDS is missing a significant portion of attacks, increasing the risk of undetected breaches and system compromise. The recall performance across models is shown in **Figure 5**.

In the non-boosting ML category, NB stood out with the highest recall after optimization (0.8837), a result of its probabilistic nature. KNN and k-Means also performed well, reaching 0.8487 and 0.8731, respectively, benefiting from their responsiveness to local data structures. RF and DT, known for their ensemble and tree-based logic, improved to 0.8828 and 0.8412, showing that ACO helped them better capture minority classes. ET, which started at 0.7752, rose to 0.8197, though it remained slightly behind due to its randomized split strategy. LR and SVM, while simpler, still showed respectable gains, ending at 0.8488 and 0.8356. Recall improvements among boosting models were consistent across the group. GB improved from 0.8342 to 0.8753, showing its

growing ability to correctly identify intrusions. AdaBoost and XGB followed similar achievements, improving to 0.872 and 0.836, respectively, both benefiting from ACO's fine-tuning of their learning processes. CatBoost, which began at a lower baseline of 0.7776, advanced to 0.8219, suggesting that its categorical feature handling became more effective after optimization. LightGBM, though still trailing with a final recall of 0.7989, showed the largest absolute gain in this group, indicating that ACO helped mitigate its initial tendency to overlook minority attack instances. Among DL models, MLP achieved the highest recall after optimization, reaching 0.8663, with its dense architecture effectively capturing attack patterns. RNN and GRU followed, recording recalls of 0.8746 and 0.8122, respectively, both benefiting from their ability to exploit temporal dependencies. CNN, while primarily spatially focused, improved to 0.7979, and LSTM advanced from 0.7751 to 0.8191, demonstrating that even deeper recurrent structures gained from ACO's tuning. Despite their architectural differences, all deep learning models exhibited upward movement, confirming the consistent impact of optimization on enhancing intrusion detection performance.

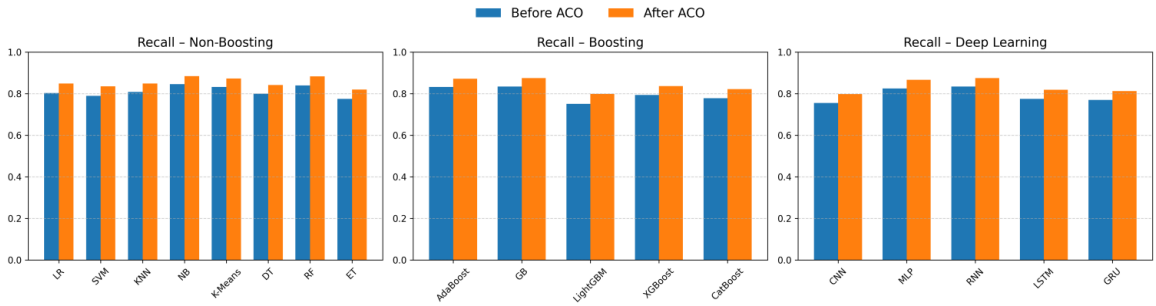


**Figure 5.** Recall comparison of the models.

## 4.5. F1-Score

The F1-Score provides a trade-off between correctly detecting attacks and not issuing false alarms. In WSNs, a high F1-score suggests the IDS is both accurate and sensitive and is able to catch intrusions while avoiding false alarms. A low F1-score reveals an imbalance, where either too many threats are missed or too many benign activities are misclassified. **Figure 6** displays the F1-score distribution, capturing the balance between detection and precision.

Among non-boosting ML models, NB obtained the highest results, with its F1-score increasing from 0.8824 to 0.9117. This reflects its consistent ability to make confident predictions while minimizing both types of error. KNN followed closely, improving from 0.8617 to 0.8939, a result of its responsiveness to local data patterns. DT and RF also showed strong post-optimization scores (0.8819 and 0.8837), suggesting that tree-based learners benefited from ACO's parameter tuning, especially in handling borderline cases. ET, which began at 0.8313, reached 0.8663, a solid gain for a model that often struggles with consistency due to its randomized splits. Meanwhile, LR and SVM, though simpler in design, achieved respectable scores of 0.8689 and 0.8648 after optimization. In the boosting category, GB delivered the highest F1-score, rising from 0.8728 to 0.9038 through its iterative refinement and resistance to overfitting. AdaBoost and XGB followed with final scores of 0.883 and 0.8723, respectively, both showing that ACO helped them sharpen their decision boundaries without sacrificing sensitivity. CatBoost, while starting lower at 0.8148, improved to 0.8512, reflecting a meaningful enhancement in its ability to handle categorical features more decisively. LightGBM, though still trailing with a final score of 0.8477, made a notable improvement from 0.8087, indicating that ACO helped mitigate its earlier imbalance between false positives and missed detections. In the DL group, all models showed consistent upward movement, with MLP offering the most stable and confident classification behavior. MLP led with an F1-score of 0.8983, up from 0.8668, due to its dense architecture appearing to have internalized the feature space effectively. RNN and GRU followed by final scores of 0.8789 and 0.8632, respectively, demonstrating that temporal modeling can yield balanced predictions when tuned properly. CNN, with its spatial focus, improved to 0.8489, while LSTM rose from 0.8142 to 0.8504, suggesting that deeper sequence models benefited from ACO's guidance despite their sensitivity to hyperparameters.

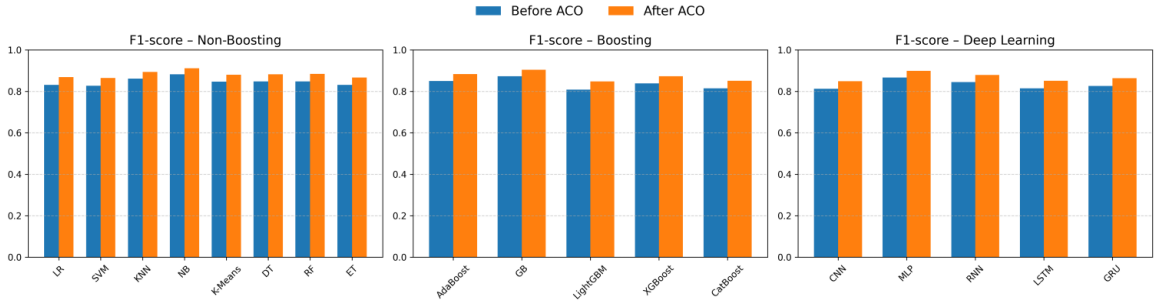


**Figure 6.** F1-score of the models.

## 4.6. Inference Efficiency

We measure inference efficiency per sample to determine how fast each model can classify one incoming packet, which is a critical factor for deploying IDS models in resource-constrained WSN environments. Since sensor nodes operate under strict energy and processing limitations, models with low training cost and fast inference are far more suitable for real-time intrusion detection. In contrast, models with higher computational demands are better suited for cluster-head or gateway devices. To evaluate this practical feasibility, **Figure 7** presents the estimated training times before and after optimization along with inference latencies for non-boosting, boosting, and deep learning models.

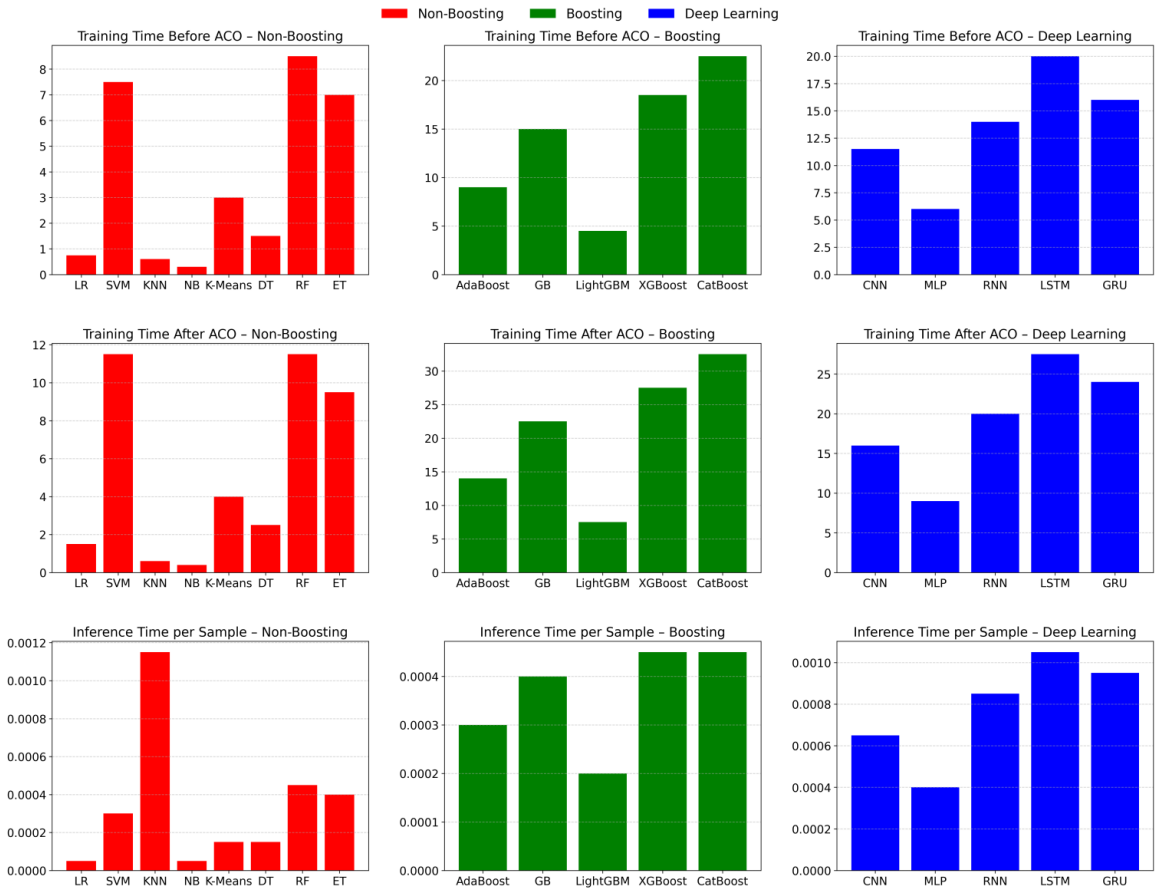


**Figure 7.** Training and inference times before and after ACO optimization.

Across all experiments, the non-boosting models form the most computationally efficient group in terms of training cost. Within this category, NB, KNN, and LR train the fastest because they rely on simple statistical or distance-based for-

mulations that require minimal parameter learning. The k-means and DT follow as moderately efficient models, reflecting their need for iterative clustering or tree construction while still remaining lightweight. In contrast, RF and ET require substantially more time due to their multi-tree ensemble structures, and SVM records the highest training time among all non-boosting models because margin optimization and kernel evaluation are computationally intensive. The boosting models represent the medium-cost group. Their training time is consistently higher than non-boosting models because boosting relies on sequential tree refinement. Among them, LightGBM is the most efficient due to its histogram-based learning strategy, while AdaBoost and GB require more iterations to converge. XGBoost and CatBoost are the most computationally demanding within this group, reflecting their deeper trees, regularization mechanisms, and more complex split-finding procedures. Deep learning models form the slowest group to train. MLP is the most efficient because of its shallow feed-forward architecture. CNN requires additional computation due to convolutional operations, while RNN, GRU, and LSTM incur the highest training cost because recurrent architectures process sequences step-by-step. LSTM and GRU are the slowest overall, consistent with their gating mechanisms and larger parameter sets.

Moreover, inference efficiency follows a similar but not identical pattern. The non-boosting models again provide the lowest latency, with NB and LR offering the fastest predictions due to their linear and probabilistic structures. K-Means, DT, and SVM remain efficient, with SVM slightly slower because of its margin-based decision function. RF and ET introduce additional overhead from ensemble voting but still remain moderate compared to more complex model families. KNN is the slowest non-boosting model, as it must compute distances to stored samples at prediction time. Boosting models form the middle tier for inference. LightGBM is the fastest because of its optimized tree layout and leaf-wise growth strategy. AdaBoost and GB remain efficient but slower, while XGBoost and CatBoost have the highest latency within this group due to deeper trees and more complex decision paths. Deep learning models exhibit the highest inference latency. MLP again performs best because of its simple forward pass. CNN requires more computation due to convolutional filters, while RNN, GRU, and LSTM are the slowest because recurrent architectures process inputs sequentially. LSTM and GRU show the highest latency, reflecting their heavier gating operations and larger parameter counts.

# 5. Conclusions

This work presents an intelligent IDS framework to protect WSN environments against a range of threats, including blackhole, grayhole, flooding, and scheduling attacks. The IDS leverages a diverse set of machine learning models, encompassing both boosting and non-boosting techniques, as well as deep learning architectures with sequential and non-sequential designs. This diversity enables the IDS to capture different traffic behaviour and decision boundaries. To further enhance detection capabilities, the ACO is integrated as a metaheuristic tuning layer across all the models in the IDS framework. Each model is implemented and evaluated to identify its strengths and limitations in the context of WSN-specific threat detection. Comparative evaluations between the baseline and ACO optimized models demonstrate consistent improvements, confirming ACO's effectiveness in strengthening the detection capability. While all four groups of classifiers benefited, the degree of improvement varied depending on the model architecture. In the non-boosting group, NB proved to be the most reliable, demonstrating strong stability and confidence in its predictions, while KNN also adapted effectively to complex and irregular data patterns. Tree-based learners followed, benefiting from optimization that reduced noise sensitivity and improved their handling of borderline cases. Even simpler models, such as LR and SVM, became more robust after tuning, showing that ACO enhanced performance across a wide range of classifiers. Within the boosting group, GB stood out as the most dependable, maintaining balanced predictions and demonstrating the clearest gains from optimization. Other boosting methods were also improved by ACO, though GB's iterative refinement gave it a distinct advantage in overall consistency. In the deep learning group, distinguishing between non-sequential and sequential architectures revealed further insights. The MLP led the non-sequential models, offering stable and confident classification behaviour, while RNN showed the strongest balance among the sequential models. CNN improved meaningfully but remained more sensitive to structural assumptions, whereas LSTM and GRU

benefited from ACO's tuning despite their architectural complexity. This framework can be deployed in real WSNs by running lightweight models on sensor nodes or cluster heads, while more complex models operate on gateway or edge devices. Feature extraction and intrusion decisions are handled hierarchically, ensuring low overhead and compatibility with typical WSN hardware constraints. Future work can extend the IDS framework by implementing reinforcement learning (RL) approaches to determine their effectiveness on WSN protection compared to ML and DL models.

## Author Contributions

Methodology, M.M. and M.F.; framework design, M.M. and M.F.; development of AI models, M.M. and M.F.; software implementation and coding, M.M. and M.F.; data analysis, M.M.; writing—original draft preparation, review, and editing: M.M. Both authors have read and agreed to the published version of the manuscript.

## Funding

This research was conducted without any external funding support.

## Institutional Review Board Statement

This study relies exclusively on computer-based simulations and does not involve human participants or animal subjects. Therefore, Institutional Review Board approval was not required. All relevant ethical standards were observed throughout the research process.

## Informed Consent Statement

Not applicable. The study does not involve human participants or animal subjects and is based solely on computational simulation methods.

## Data Availability Statement

The results obtained in this study can be made publicly available upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## AI Use Statement

During the preparation of this work, the authors used the Grammarly extension in Microsoft Word in order to refine the language. After using this tool, the authors reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

## References

[1] Nguyen, M.D., Nguyen, M.T., Nguyen, D.T., et al., 2025. A Comparative Study of Wi-Fi Technologies in Wireless Sensor Networks. Computer Networks and Communications. 3(1), 75–87. DOI: https://doi.org/10.37256/cnc.3120256070

[2] Sicari, S., Rizzardi, A., Miorandi, D., et al., 2018. REATO: REActing TO Denial of Service Attacks in the Internet of Things. Computer Networks. 137, 37–48. DOI: https://doi.org/10.1016/j.comnet.2018.03.020

[3] Altigani, A., Hasan, S., Barry, B., et al., 2021. A Polymorphic Advanced Encryption Standard—A Novel Approach. IEEE Access. 9, 20191–20207. DOI: https://doi.org/10.1109/ACCESS.2021.3051556

[4] Altulaihan, E., Almaiah, M.A., Aljughaiman, A., 2024. Anomaly Detection IDS for Detecting DoS Attacks in IoT Networks Based on Machine Learning Algorithms. Sensors. 24(2), 713. DOI: https://doi.org/10.3390/s24020713

[5] Fares, H., Vibhute, A., Mouniane, Y., et al., 2025. Intrusion Detection in Wireless Sensor Networks Using Machine Learning. Procedia Computer Science. 252, 912–921. DOI: https://doi.org/10.1016/j.procs.2025.01.052

[6] Mahadik, S.S., Pawar, P.M., Muthalagu, R., 2023. Edge-HetIoT Defense Against DDoS Attack Using Learning Techniques. Computers & Security. 132, 103347. DOI: https://doi.org/10.1016/j.cose.2023.103347

[7] Sharma, D.K., Dhurandher, S.K., Kumaram, S., et al., 2022. Mitigation of Black Hole Attacks in 6LoWPAN RPL-Based Wireless Sensor Networks for Cyber Physical Systems. Computer Communications. 189, 182–192. DOI: https://doi.org/10.1016/j.comcom.2022.04.003

[8] Nordin, N., Mohd Pozi, M.S., 2024. Cross-Layer Based Intrusion Detection System for Wireless Sensor Net-

works: Challenges, Solutions, and Future Directions. In: Zakaria, N.H., Mansor, N.S., Husni, H., et al. (Eds.). Computing and Informatics (ICOCI 2023): Communications in Computer and Information Science, Vol 2001. Springer: Singapore. pp. 108–121. DOI: https://doi.org/10.1007/978-981-99-9589-9_9

[9] Ramesh, S., Yaashuwanth, C., Prathibanandhi, K., et al., 2021. An Optimized Deep Neural Network-Based DoS Attack Detection in Wireless Video Sensor Networks. Journal of Ambient Intelligence and Humanized Computing. DOI: https://doi.org/10.1007/s12652-020-02763-9

[10] Liu, G., Zhao, H., Fan, F., et al., 2022. An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs. Sensors. 22(4), 1407. DOI: https://doi.org/10.3390/s22041407

[11] El Kouari, O., Lazaar, S., Achoughi, T., 2025. Fortifying Industrial Cybersecurity: A Novel Industrial Internet of Things Architecture Enhanced by Honeypot Integration. International Journal of Electrical and Computer Engineering. 15(1), 1089–1098. DOI: https://doi.org/10.11591/ijece.v15i1.pp1089-1098

[12] Gebreyesus, G., Panda, J., Sreedevi, I., 2023. Localization and Detection of Multiple Attacks in Wireless Sensor Networks Using Artificial Neural Network. Wireless Communications and Mobile Computing. 2023(1), 2744706. DOI: https://doi.org/10.1155/2023/2744706

[13] Osanaiye, O., Alfa, A.S., Hancke, G.P., 2018. A Statistical Approach to Detect Jamming Attacks in Wireless Sensor Networks. Sensors. 18(6), 1691. DOI: https://doi.org/10.3390/s18061691

[14] Almomani, I., Alenezi, M., 2018. Efficient Denial of Service Attacks Detection in Wireless Sensor Networks. Journal of Information Science and Engineering. 34(4), 977–1000. DOI: https://doi.org/10.6688/JISE.201807_34(4).0011

[15] Elsadig, M.A., 2023. Detection of Denial-of-Service Attack in Wireless Sensor Networks: A Lightweight Machine Learning Approach. IEEE Access. 11, 83537–83552. DOI: https://doi.org/10.1109/ACCESS.2023.3303113

[16] Vinayakumar, R., Alazab, M., Soman, K.P., et al., 2019. Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access. 7, 41525–41550. DOI: https://doi.org/10.1109/ACCESS.2019.2895334

[17] Shanmathi, M., Sonker, A., Hussain, Z., et al., 2024. Enhancing Wireless Sensor Network Security and Efficiency with CNN-FL and NGO Optimization. Measurement: Sensors, 32, 101057. DOI: https://doi.org/10.1016/j.measen.2024.101057

[18] Al Sukkar, G., Al-Sharaeh, S., 2025. Enhancing Security in Wireless Sensor Networks: A Machine Learning-Based DoS Attack Detection. Engineering, Technology & Applied Science Research, 15(1), 19712–19719. DOI: https://doi.org/10.48084/etasr.7191

[19] Baswaraj, D., Palanikumar, S., Gopalakrishnan, T., et al., 2025. Enhancing Security in Energy-Efficient Wireless Sensor Networks Using Deep Learning. International Journal of Sensors, Wireless Communications and Control, 15(4), 305–315. DOI: https://doi.org/10.2174/0122103279327259240829041329

[20] Wakili, A., Bakkali, S., Alaoui, A.E.H., 2024. Machine Learning for QoS and Security Enhancement of RPL in IoT-Enabled Wireless Sensors. Sensors International, 5, 100289. DOI: https://doi.org/10.1016/j.sintl.2024.100289

[21] Karthikeyan, M., Manimegalai, D., RajaGopal, K., 2024. Firefly Algorithm-Based WSN-IoT Security Enhancement with Machine Learning for Intrusion Detection. Scientific Reports. 14(1), 231. DOI: https://doi.org/10.1038/s41598-023-50554-x

[22] Wang, Z., Ding, H., Li, B., et al., 2022. Energy-Efficient Cluster-Based Routing Protocol for WSN Using Firefly Algorithm and Ant Colony Optimization. Wireless Personal Communications, 125, 2167–2200. DOI: https://doi.org/10.1007/s11277-022-09651-9

[23] Tawfeek, M.A., Alrashdi, I., Alruwaili, M., et al., 2025. Improving Energy Efficiency and Routing Reliability in Wireless Sensor Networks Using Modified Ant Colony Optimization. EURASIP Journal on Wireless Communications and Networking, 2025, 22. DOI: https://doi.org/10.1186/s13638-025-02449-w

[24] Razooqi, Y.S., Al-Asfoor, M., Abed, M.H., 2024. Optimize Energy Consumption of Wireless Sensor Networks Using Modified Ant Colony Optimization (ACO). Networking and Internet Architecture. arXiv preprint. arXiv:2402.12526. DOI: https://doi.org/10.48550/arXiv.2402.12526

[25] Kooshari, A., Fartash, M., Mihannezhad, P., et al., 2024. An Optimization Method in Wireless Sensor Network Routing and IoT with Water Strider Algorithm and Ant Colony Optimization. Evolutionary Intelligence, 17, 1527–1545. DOI: https://doi.org/10.1007/s12065-023-00847-x

[26] Almomani, I., Al-Kasasbeh, B., Al-Akhras, M., 2016. WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks. Journal of Sensors. 2016(1), 4731953. DOI: https://doi.org/10.1155/2016/4731953

[27] Kaggle, 2024. WSN-DS: A dataset for intrusion detection systems in wireless sensor networks. Available from: https://www.kaggle.com/datasets/bassamkasasbeh1/wsnds (cited 12 January 2026).