DOI: 10.6977/IJoSI.202510_9(5).0005
R. Gayathri, K.N. Shreenath, etc./Int. J. Systematic Innovation, 9(5), 56-70 (2025)

Advanced fault detection in wireless sensor networks: A metaheuristic-driven deep learning approach to enhance the quality of service

R. Gayathri*, K. N. Shreenath

Department of Computer Science and Engineering, Siddaganga Institute of Technology, Visvesvaraya Technological University, Tumakuru, Karnataka, India

*Corresponding author E-mail: gayathrirkrishna@gmail.com

(Received 20 December 2024; Final version received 10 June 2025; Accepted 28 August 2025)

Abstract

Wireless sensor networks (WSNs) face critical challenges in fault detection that can compromise their quality of service in dynamic environments. This study introduces an integrated framework that enhances fault detection by combining advanced noise filtering, optimized feature selection, and a robust deep learning (DL) model. The framework employs a dynamic noise filtering technique with adaptive thresholding to effectively remove noise while preserving essential data integrity. Complementing this, the rank-based whale optimization algorithm refines feature selection, boosts model performance, and reduces computational demands. At its core, the hierarchical attention-based DL model utilizes temporal convolutional layers, long short-term memory units, and hierarchical attention mechanisms to capture both short-term and long-term dependencies in the data. Experimental evaluations on the WSN dataset demonstrate outstanding performance, with a precision of 0.98, a recall of 0.99, an F1-score of 0.98, and an area under the curve of 0.99 for all fault classes. Comparative analysis reveals that this framework outperforms existing approaches in terms of accuracy, sensitivity, specificity, and computational efficiency. Overall, the proposed solution improves fault detection and enhances network reliability, minimizes false alarms, and extends the operational lifespan of WSNs, offering a scalable approach for mission-critical applications in healthcare, environmental monitoring, and industrial automation.

Keywords: Dynamic Noise Filtering, Hierarchical Attention-based Deep Learning, Long Short-term Memory, Quality of Service, Rank-based Whale Optimization Algorithm, Wireless Sensor Networks

1. Introduction

Wireless sensor networks (WSNs) are a game-changing technology that allows gathering, processing, and sending data from dispersed sensor nodes. These nodes can perceive and monitor their surroundings since they are outfitted with various sensors and communication tools (Gebremariam et al., 2023). Environmental assessment, smart cities, healthcare, and industrial automation are just a few industries that use WSNs. Their capability to gather data remotely and in real-time from inaccessible or dangerous regions enables effective data-driven decision-making (Chataut et al., 2023; Talukder et al., 2024). The sensitive nature of data being transferred and the possibility of network flaws make WSN security crucial (Yakubu and Maiwada, 2023). Data manipulation, denial of service attacks, and illegal access are just a few security risks that WSNs face (Nimbalkar et al., 2023). These dangers are more likely to affect WSNs because of their dispersed and wireless nature. Protecting the privacy, availability, and integrity of data in WSNs is essential to preserving these networks' credibility and dependability (Alghamdi et al., 2023). The goal of intrusion detection, a crucial part of WSN security, is to identify and stop harmful activity on the network (Heidari and Jabraeil, 2022). Conventional rule-based intrusion detection systems frequently use predefined signatures or criteria, which are ineffective in identifying more complex assaults (Sezgin and Boyaci, 2022). One method that has shown promise

for WSN intrusion detection is machine learning (ML). ML algorithms provide proactive and adaptive security measures by learning from past data and spotting abnormalities or patterns suggestive of possible breaches (Talukder et al., 2023). Intelligent intrusion detection systems may be developed in WSN owing to ML techniques. These algorithms can distinguish between benign and malicious behavior, analyze vast volumes of data, and identify odd trends (Ghazal, 2022). By extracting useful information from intricate WSN datasets, ML techniques such as decision trees, random forests, neural networks, and gradient boosting techniques can increase the precision and efficacy of intrusion detection systems (Talukder et al., 2022).

Internet of Things (IoT) systems have unique characteristics, such as restricted bandwidth capacity (Qaiwmchi et al, 2020), limited energy, heterogeneity, global connection, and ubiquity, which make typical intrusion detection system solutions inadequate or less effective for their security. Deep learning (DL) and ML-related approaches have earned a reputation for their efficacious use in identifying network vulnerabilities, particularly those on IoT networks (Pandey et al., 2022). WSNs do not directly employ traditional network intrusion detection methods because of their poor computing and communication capabilities. Several WSN intrusion detection researchers can currently use ML algorithms to examine traffic data. Due to the WSN network's growing user base and network size, it generates high-dimensional traffic data. Traditional ML models struggle with low feature extraction and detection accuracy, making them unsuitable for an application environment (Almomani, 2021). The detection model's precision can be increased using DL instead of ML models for intrusion detection systems, as they can learn the data flow features and reduce the computational load (Sharmin et al., 2023). This study aims to develop an integrated fault detection framework for WSNs that improves data reliability and overall network performance under dynamic conditions. The framework is designed to address challenges such as noise interference and high computational demands in fault detection based on the following contributions:

(i) Introduces a dynamic noise filtering (DNF) technique with adaptive thresholding to remove noise from sensor data while preserving critical information
(ii) Utilizes the rank-based whale optimization algorithm (RWOA) to select the most relevant and non-redundant features, thereby boosting model performance and reducing computational complexity
(iii) Develops a hierarchical attention-based DL (HADL) model that integrates temporal convolutional layers, long short-term memory (LSTM) units, and hierarchical attention mechanisms to capture both short-term and long-term dependencies in the data, leading to superior fault detection accuracy
(iv) Demonstrates exceptional performance on the WSN dataset (WSN-DS) with precision, recall, F1-scores, and area under the curve (AUC) values of 0.99 or higher, outperforming existing methods in accuracy, sensitivity, specificity, and computational efficiency.

This study provides a systematic overview of a research project addressing fault detection challenges in WSNs. It begins with an introduction; reviews existing studies; proposes a novel framework integrating noise filtering, feature optimization, and a hierarchical DL model; compares the approach against existing methodologies; and concludes with key contributions and potential future directions.

2. Related Work

The literature survey section provides a comprehensive overview of existing approaches in fault detection for WSNs. It examines the evolution of techniques in noise filtering, feature selection, and DL, identifying the strengths and limitations of current methodologies.

Tan et al. (2019) introduced an intrusion detection approach that leverages a random forest classifier enhanced by the synthetic minority oversampling technique to address dataset imbalance, improving accuracy from 92.39% to 92.57%. In a similar vein, Rezvi et al. (2021) employed a data mining framework to discern various types of denial of service attacks by comparing several classifiers—such as K-nearest neighbors (KNN), naïve Bayes, logistic regression, support vector machine, and artificial neural network—with their findings indicating that artificial neural network and KNN yielded superior accuracies of 98.56% and 98.4%, respectively. Meng et al. (2022) proposed an intrusion detection method tailored for resource-constrained WSNs, integrating a light gradient boosting machine with recursive feature elimination, Shapley additive explanations analysis, and an iterative tree model, in combination with the synthetic minority oversampling technique-Tomek balancing technique, which resulted in detection rates exceeding 99% for all attack types and a substantial reduction (46%) in modeling time.

Singh et al. (2020) developed a fuzzy rule-based intrusion prevention system that classifies sensor nodes into risk categories based on metrics, like packet delivery ratio, energy consumption, and signal strength, achieving an accuracy of 98.29% and effectively neutralizing malicious nodes. Alruhaily et al. (2021) designed a multi-tier intrusion detection architecture

incorporating a real-time naïve Bayes classifier at the network edge and a cloud-based random forest classifier for comprehensive packet analysis, with their system delivering high detection accuracies across various attack categories. Complementing these efforts, Chandre et al. (2022) employed a convolutional neural network within a DL framework to detect and prevent intrusions by extracting robust feature representations from extensive labeled datasets, reaching an accuracy rate of 97%.

Further advancing the field, an optimized collaborative intrusion detection system was proposed by Elsaid and Albatati (2020) using an updated artificial Bee colony optimization (BCO) algorithm that enhanced resource efficiency and detection accuracy while integrating a weighted support vector machine to minimize false alarms through effective coordination among base stations, cluster heads, and sensor nodes. Addressing data imbalance in WSN cyberattacks, Putrada et al. (2022) demonstrated that extreme gradient boosting outperformed decision trees and naïve Bayes by achieving the highest AUC values across multiple attack classes. Ravindra et al. (2023) introduced an innovative anomaly detection technique that utilizes data compression and dynamic thresholding, powered by an enhanced extreme learning machine coupled with an enhanced transient search arithmetic optimization (ETSAO) algorithm, which successfully reduced computational overhead and achieved a 96.90% accuracy on the WSN-DS. Finally, Alruwaili et al. (2023) presented the red kite optimization algorithm (RKOA) with an average ensemble model for intrusion detection (AEID) methodology for IoT-based WSNs, which incorporates feature selection through RKOA, min-max normalization, and an average ensemble learning model with hyperparameter tuning using a Lévy-fight chaotic whale optimization technique, resulting in an improved accuracy of 98.94%.

While current methodologies effectively address individual aspects such as detection accuracy, computational efficiency, and class imbalance, they seldom integrate noise filtering, feature selection, and DL-based fault detection into a unified framework. Moreover, many approaches do not fully exploit hierarchical DL architectures capable of capturing both short-term and long-term temporal dependencies inherent in sensor data. This gap underscores the need for a comprehensive and scalable solution that simultaneously enhances network reliability, minimizes false alarms, and extends the operational lifespan of WSNs, thereby offering robust performance in dynamic and resource-constrained environments.

3. Proposed Methodology

The proposed methodology introduces an advanced framework (Fig. 1) for fault detection in WSNs, addressing the critical challenges of noise interference, suboptimal feature selection, and inaccurate fault classification in dynamic environments. By integrating a suite of cutting-edge techniques, the approach ensures enhanced fault detection accuracy and robust network performance while optimizing computational efficiency. The methodology begins with DNF using adaptive thresholding, a real-time noise mitigation strategy that dynamically adjusts thresholds based on statistical analysis of noise patterns in sensor data. This ensures the preservation of critical fault-indicative information while filtering out irrelevant fluctuations, even under varying environmental conditions. This adaptive mechanism significantly enhances the data quality fed into the fault detection pipeline. To extract the most relevant and non-redundant features, an RWOA was utilized. This novel metaheuristic optimization approach combines the global exploration capabilities of WOA with feature relevance ranking using mutual information. By balancing classification accuracy and feature dimensionality, the RWOA ensures the selection of an optimal, compact feature set, reducing computational overhead while maintaining precision.

For fault classification, the proposed framework leverages an HADL model. This multi-layered architecture incorporates temporal convolutional layers to capture short-term patterns and anomalies, followed by LSTM layers to model long-term dependencies in time-series data. The centerpiece of HADL is its dual-level hierarchical attention mechanism, which prioritizes critical features within each time step and across the sequence, maximizing interpretability and decision accuracy. The final classification layer delivers precise fault predictions, adapting to the complexities of real-world WSN scenarios. The proposed methodology establishes a high-performance fault detection framework by synergistically combining noise filtering, feature optimization, and a DL-based classification model. This approach not only enhances the quality of service but also ensures the scalability, reliability, and efficiency of WSNs in mission-critical applications. Integrating adaptive mechanisms and optimization-driven feature selection represents a significant advancement in fault detection technology, paving the way for more resilient and intelligent WSN deployments.

3.1. DNF Technique with Adaptive Thresholding

Initially, a DNF technique with adaptive thresholding effectively filters out noise while preserving critical data for fault detection in WSNs. First, the technique continuously monitors incoming

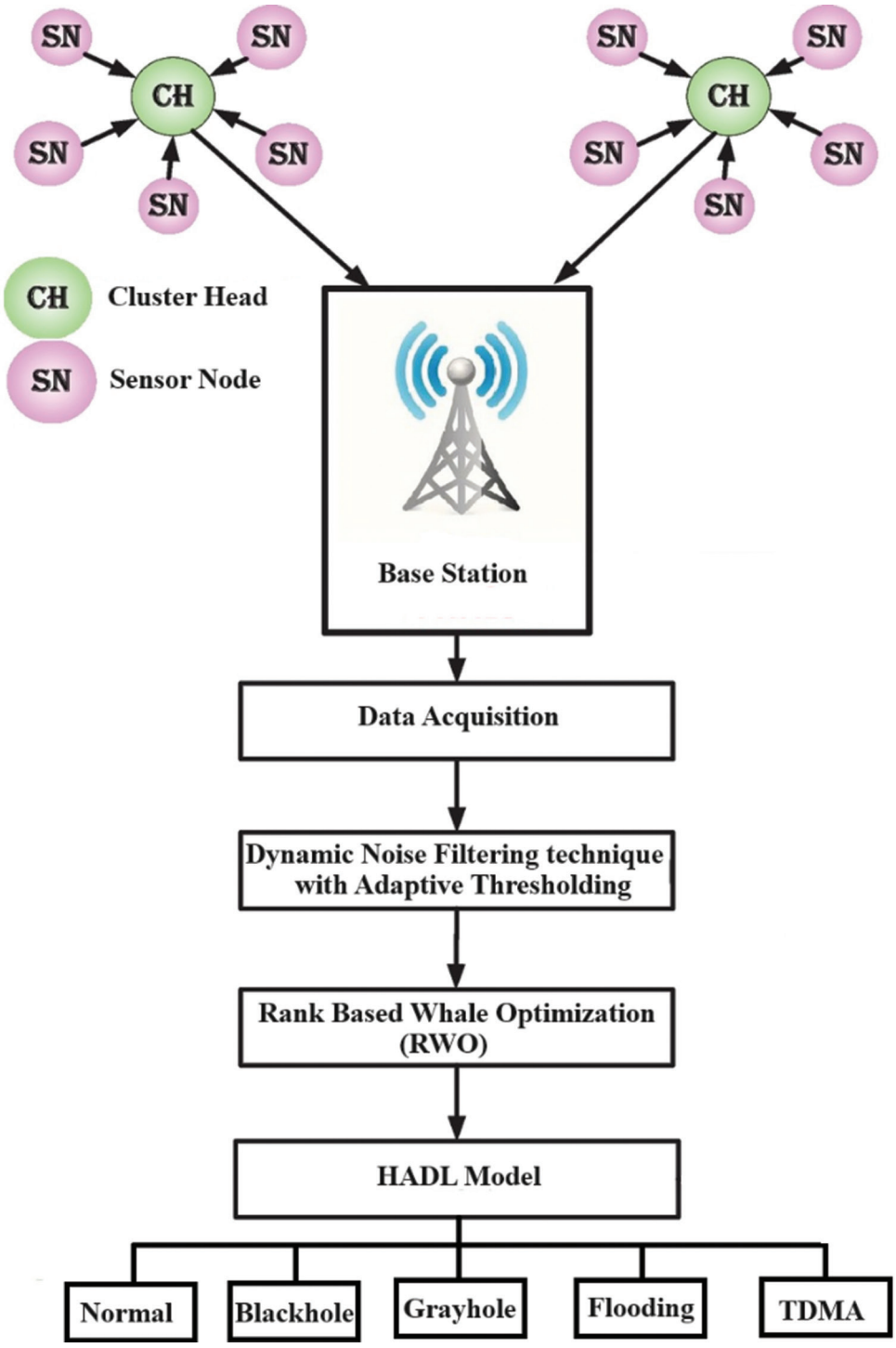

Fig. 1. Flow diagram of the proposed methodology
Abbreviations: HADL: Hierarchical attention-based deep learning; TDMA: Time-division multiple access

sensor data to assess the real-time noise levels and distribution patterns. It then calculates statistical properties, such as the mean and standard deviation of the noise, across a sliding window of recent data. Based on these calculations, the method dynamically adjusts the noise threshold, increasing it during high-noise periods to avoid false positives and lowering it when data quality improves to ensure that subtle faults are not missed. This adaptive threshold is then applied to filter out noise, allowing only data points that exceed an adjusted threshold to pass through for further processing. The process is repeated continuously, ensuring the filtering adapts to changing network conditions, resulting in a more accurate and reliable dataset for subsequent analysis.

DNF with adaptive thresholding is a technique

used to improve the data quality in systems like WSNs, where data can be corrupted by noise due to various factors like sensor malfunctions, environmental interference, or communication issues. This method enhances the signal-to-noise ratio by removing noise without losing important information. This component refers to identifying and reducing noise in the data in real-time or dynamically, based on varying conditions. The noise filtering adapts to the type of noise and the changing characteristics of the signal. In a WSN, sensor readings can be affected by several types of noise, such as random fluctuations or environmental disturbances. DNF identifies and selectively removes these anomalies in the data, ensuring that useful signals are preserved.

Thresholding involves setting a limit (threshold) above or below which the data is considered noise or valid. Adaptive thresholding adjusts this threshold based on the current state of the data. In a dynamic environment, where sensor data characteristics change over time, a static threshold might not work effectively. The adaptive threshold is recalculated periodically or based on specific criteria, such as the variance of the data, the signal strength, or statistical measures of the data distribution. For example, if the sensor data shows sudden spikes or sharp drops (indicative of noise), the threshold can be adjusted to treat these as noise and filter them out. Conversely, when data becomes more stable or predictable, the threshold can be widened to capture a broader range of valid information. The algorithm for DNF with adaptive thresholding is as follows:

(i) Step 1: The system continuously monitors incoming sensor data for unusual patterns, sudden spikes, or deviations from expected values, characteristic of noise
(ii) Step 2: The system uses adaptive techniques to determine a dynamic threshold that reflects the current data distribution, variability, or other environmental factors. The threshold changes are based on observed conditions, such as the variance of the signal or the presence of unusual outliers
(iii) Step 3: Data points outside the adaptive threshold are flagged as noise and discarded or replaced. The remaining data is preserved for further processing and analysis
(iv) Step 4: By dynamically adjusting the threshold, the method ensures that important or meaningful data is not discarded while filtering out noise. This allows for better quality input for downstream analysis, such as fault detection in WSNs.

Removing noise without discarding useful data improves the quality of sensor readings, leading to better analysis and decision-making. The adaptive threshold can adjust to different types of noise or changes in the network conditions, making it more robust in dynamic environments. In systems like fault detection, reducing noise ensures that only actual faults are detected, minimizing false alarms. In summary, DNF with adaptive thresholding ensures that the data used for analysis in WSNs or similar systems is of high quality, with noise effectively removed based on real-time conditions.

3.2. RWOA Technique

This research presents an RWOA to improve the efficacy of feature extraction. It uses the latest developments in metaheuristic optimization techniques to find the most pertinent features for defect identification. To start, the WOA searches the feature space to optimize a fitness function that strikes a compromise between feature set size and classification effectiveness. The algorithm effectively explores the search space, avoiding local optima and locating the optimal solution globally by imitating the bubble-net feeding method of humpback whales. Features with stronger correlations are given larger weights. In parallel, the dependence of each characteristic on the goal variable (i.e., defect or normal state) is evaluated using mutual information. The selected features from the WOA are then refined using the mutual information ranking, ensuring that only the most informative and non-redundant features are retained. This hybrid approach significantly improves the robustness and accuracy of the fault detection model by ensuring that the extracted features are both optimal in relevance and minimal in quantity, reducing the computational burden.

One of the popular population-based metaheuristic algorithms for solving global optimization problems in various fields is the WOA algorithm, developed by Mirjalili and Lewis (2016). The humpback whale's natural hunting behavior serves as the model for this program. At the water's surface, humpback whales hunt by focusing on schools of krill or tiny fish. To encircle and seize their prey, they form characteristic bubbles in a spiral pattern. The whales descend and swim to the water's surface, creating spiral bubbles around the prey. The WOA uses three tactics to mimic whale behavior: (i) spiral bubble-net attacking (exploitation phase), (ii) hunting for prey (exploration phase), and (iii) surrounding the target. $X_i^t = \left(x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,D}^t\right)$ represents the location of the i^{th} whale at iteration t, where $i = 1,2,\ldots, N$ and N and D represent the whale population and the problem's dimensions, respectively.

3.3. Encircling Prey Strategy

Whales can track down and enclose their prey. The ideal choice for whales in WOA is the target prey

or a nearby location inside the search area. Eq. (1) is used by other whales to update their location as they try to approach the ideal agent during prey encirclement. This equation is constructed with t representing the current iteration, X_i^t representing the i^{th} whale's location for the current iteration, and X^{*t} representing the position vector of the best solution, thus far, which is updated in each iteration if a better solution is discovered.

$$X_i^{t+1} = X^{*t} - A \cdot D \tag{1}$$

$$D = \left| C \times X^{*t} - X_i^t \right| \tag{2}$$

where D stands for the distance between the whale and the prey X^{*t}, which is established by Eq. (2). I denote the absolute value, A and C represent coefficient vectors that are established using Eqs. (3) and (4).

$$A = 2 \times a \times r - a \tag{3}$$

$$C = 2 \times r \tag{4}$$

$$a = 2 - t \times \left(\frac{2}{MaxIter} \right) \tag{5}$$

According to Eq. (5), the parameter r in Eqs. (3) and (4) are random numbers in the interval, whereas Eq. (3) declines linearly from 2 to 0 repetitions. The values t and Max_{iter} are used in Eq. (5) to represent the current iteration and the total number of iterations. Through the use of the parameter a, the whales are gradually brought into the surrounding scope.

3.4. Spiral Bubble-net Attacking Strategy

Humpback whales use a bubble net to spiral toward their prey and corner them. Two strategies are used to mathematically represent this strategy: spiral updating position and shrinking encircling.

3.4.1. Shrinking encircling method

In Eq. (3), this tendency is reflected by reducing the value of the convergence variable a. Furthermore, the alternate range of A fluctuation is linearly lowered from 2 to 0, utilizing the parameter a through iterations. In other words, A is a random value belonging to the interval $[-a,a]$.

3.4.2. Spiral updating position method

First, this method calculates the distance between whales X_i^t using Eq. (6); X^{*t} is the best result thus far. Next, a spiral migration from its present location towards an ideal solution is described using Eq. (7). In these calculations, the logarithmic spiral shape is determined by a constant parameter, b, and a random variable, l, between [−1,1].

$$D' = \left| X^{*t} - X_i^t \right| \tag{6}$$

$$X_i^{t+1} = D' \times e^{bl} \times \cos(2\pi l) + X^{*t} \tag{7}$$

The logarithmic spiral form is determined by a fixed parameter, b and a random value, l, that falls between [−1,1]. The humpback whale swims in WOA, spiraling around its prey in a tight circle. The spiral model or the diminishing encircling method is the two options the whale chooses for changing its location during the optimization phase. The mathematical model is defined by Eq. (8), where p is a random number in [0, 1].

$$X_i^{t+1} = \begin{cases} X^{*t} - A \times D \ \ if \ \ p < 0.5 \\ D' \times e^{bl} \times \cos(2\pi l) + X^{*t} \ \ if \ \ p \geq 0.5 \end{cases} \tag{8}$$

3.5. Searching for Prey Strategy

Whales employ this strategy to increase population diversity and seek the problem space for uncharted territory. A randomly selected search agent updates the position of each whale. To avoid being caught in a local minimum, the search agent is pushed away from a randomly chosen humpback whale using the parameter A. Eq. (9) is employed for exploration [31].

$$\begin{aligned} X_i^{t+1} &= X_{rand} - A \times D \\ D &= \left| C \times X_{rand} - X_i^t \right| \end{aligned} \tag{9}$$

here A and C are calculated using Eqs. (3) and (4), and X_{rand} is a random position vector in the search space chosen from the available whales in the population.

After N, when whales are randomly distributed over the search space, the association objective function value is determined, as seen in the WOA flowchart in Fig. 2. When the initial values of the control parameters

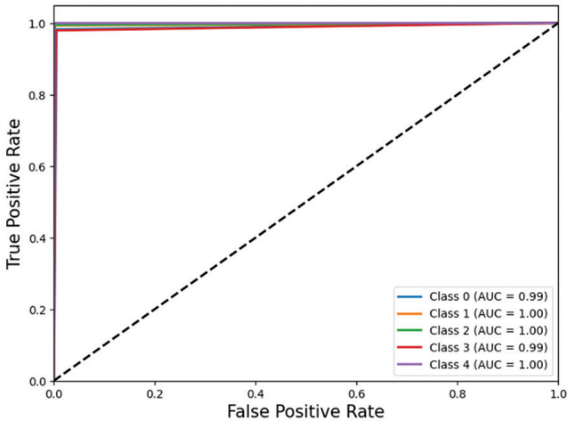

Fig. 2. Area under the curve (AUC) of the classes

are altered, the optimization begins with the present iterations. At each iteration, the value of the parameter p is then evaluated. Eq. (7) specifies the spiral updating position method used by whales when $p \geq 0.5$. Whales update their location when $p < 0.5$ using the encircling prey strategy (Eq. [1]) if $|A|<1$ and the hunting for prey strategy (Eq. [9] if $|A|\geq 1$). Subsequently, the fitness and viability values of the newly gained positions are computed. Next, the ideal solution is updated, and WOA is ultimately ended.

The WOA is based on humpback whale hunting behavior, mimicking the bubble-net strategy. It involves exploration and exploitation, with whales randomly moving in search of space and using shrinking encircling mechanisms and spiral movements. However, WOA can face challenges like premature or slow convergence due to poor exploration and exploitation balance. The rank-based method is introduced in RWOA to enhance the population selection mechanism during the optimization process. This method changes how whales are selected for exploration and exploitation by considering their rank in the population rather than selecting them randomly or with equal probability.

3.5.1. Rank assignment

After evaluating the fitness of all candidates (whales), they are ranked based on their fitness values (i.e., solutions with lower objective function values are ranked higher if the goal is minimization). Each individual in the population is assigned a rank based on their fitness, with the best solution (with the lowest fitness) getting rank 1, the second-best getting rank 2, and so on.

3.5.2. Probability-based selection

Instead of choosing individuals to update their position randomly or based on fixed probabilities, rank-based selection assigns higher probabilities to individuals with better (lower) ranks. The better individuals (those with better fitness) are more likely to be selected for the exploitation phase, while the worse individuals are more likely to be selected for the exploration phase. A non-linear probability distribution is often used, so the probability of selecting a whale is inversely proportional to its rank. This ensures that the algorithm focuses more on promising solutions while maintaining some diversity by allowing worse solutions to participate in the search process.

3.5.3. Exploration and exploitation

During the exploration phase, the worst-ranked individuals (those with higher ranks) can move freely, encouraging the algorithm to explore a wide area of the search space. During the exploitation phase, the best-ranked individuals (those with lower ranks) are more likely to contribute to the search, refining the solution by focusing on regions with promising results.

3.5.4. Fitness-based movement

The movement of each whale is influenced by its rank. For example, for better individuals (lower ranks), they will likely refine their position by getting closer to the best solution. For worse individuals (higher ranks), they are more likely to perform a broader search to avoid premature convergence and encourage diversity.

3.5.5. Rank-based update of positions

The whale's position update rule, which typically involves a spiral or encircling mechanism, can also be influenced by the whale's rank. For example, whales with better ranks (i.e., better solutions) may use the shrinking encircling method with higher probabilities to exploit reasonable solutions, while whales with worse ranks can have a higher probability of using random search to explore new regions of the search space.

The RWOA improves convergence by ensuring better solutions drive the search process, leading to faster and more accurate results. It enhances diversity by allowing worse solutions to explore the search space, avoiding premature convergence, and maintaining population diversity. RWOA also balances exploration and exploitation, allowing for a broader search space and reducing the risk of stagnation by encouraging weaker solutions to explore new areas.

3.6. Hierarchical Attention-based DL Model

Finally, a HADL model is employed for fault detection in WSNs, which starts with an embedding layer that converts the raw input features from the WSN-DS into dense vectors, capturing the underlying patterns in a compressed form. Following this, temporal convolutional layers detect patterns and anomalies over short data sequences, focusing on how features change over time. These layers help identify sudden shifts or unusual trends that might indicate faults. Next, recurrent layers, such as LSTM units, capture long-term dependencies in the time-series data, effectively modeling how earlier data points influence future observations. The central innovation of HADL is its hierarchical attention mechanism, which is applied at multiple stages: first, to highlight the most relevant features within each time step, and then to focus on the most important time steps across the sequence. This dual-level attention ensures the model

prioritizes the most critical information for accurate fault detection. Finally, the processed data is passed through a fully connected layer, which integrates the information from all previous layers and leads to a softmax classification layer. This final layer provides a probability distribution over the fault classes, allowing the model to make precise and confident predictions about the network's state. This layered structure of HADL ensures that the model effectively captures both immediate and long-term patterns in the data, leading to enhanced accuracy in detecting faults in WSNs.

The rank-based selection in RWOA improves convergence by ensuring better solutions drive the search process, leading to faster and more accurate results. It enhances diversity by allowing worse solutions to explore the search space, avoiding premature convergence, and maintaining population diversity. RWOA also balances exploration and exploitation, allowing for a broader search space and reducing the risk of stagnation by encouraging weaker solutions to explore new areas.

Recurrent neural networks are widely known for their ability to capture the dynamics of sequential data when working with time-sequence data supplied by monitoring systems. In contrast to a traditional neural network, HADL neurons are reinforced by including edges that span neighboring time steps. These links, which are referred to as recurrent edges, create cycles that are self-connected of a neuron to itself over time, adding a temporal component to the model data space. The behavior of a neuron with recurrent edges in a basic recurrent network may be explained as follows in Eq. (10):

$$h^{(t)} = F(Wh^{(t-1)} + Ux^{(t)} + b_h)$$
$$y^{(t)} = G(Vh^{(t)} + b_y) \quad (10)$$

where $h^{(t)}$ represents the hidden layer activation at time t, $h^{(t-1)}$ represents the previously hidden representation, and x (t) represents the input layer's current input. The input-to-hidden, hidden-to-hidden, and hidden-to-output connections are parametrized by the weight matrices W, U, and V, respectively, within the HADL. The output layer and hidden layer bias parametersb_y and b_h allow offset learning. The two layers' activation functions are F and G, respectively. The recurrent neural network's output is $y^{(t)}$. In contrast to the propagation between layers, which is cyclic, the data propagation is one-way in the time direction when the network is unfurled from left to right. The distinction lies in the weights (W) being shared between time steps. The network may therefore be trained across several time steps using a backpropagation approach. As t2–t1 grows in size, the input's contribution to time step t2 will either move to infinity or decay to zero since the weights are the same for all time steps. The loss gradient will also either burst or decay to the input, depending on the activation function f and whether $|W| > 0$ *or* $|W|<0$.

In the HADL method, every neuron in the hidden layer is substituted by a memory cell architecture with a core node known as the state unit $\boldsymbol{s}(^t)$. This model's architecture is similar to that of a typical recurrent neural network with a hidden layer. Like a typical neuron in a hidden layer, the cell has external outputs to the next time step and the layer below, as well as external inputs from the previous layer and the prior state. In addition, it features an internal set of gating units that use multiplication to regulate the information flow. Updates are made to the forgetting gate unit $f_i^{(t)}$, state unit $s^{(t)}$, input gate unit $g_i^{(t)}$, output gate unit $q_i^{(t)}$, and output $h_i^{(t)}$ for each time step t based on the current input $x_j^{(t)}$ and the prior output $h_j^{(t-1)}$. Below is the computing process for an LSTM model at each stage (Eq. [11]):

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right)$$

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)} \quad (11)$$

The state unit and the three gate units are all triggered by the sigmoid function σ (·) and have their own bias b_i, input weights $U_{i,j}$, and recurrent weights $W_{i,j}$. Ultimately, the HADL cell's output is modified to reflect the hidden layer vector $h_i^{(t)}$. When an input/output gate is activated in the forward direction, the HADL may learn when and to what degree to let values in/out. The value of the hidden layer will neither increase nor decrease if both gates are closed, meaning that neither outputs nor intermediate time steps will be impacted. The gradients can also propagate backward throughout many time steps without disappearing or bursting. That is, gates may learn when to allow error to enter and when to limit it. The ability of HADL to learn long-term dependencies more effectively than standard recurrent designs has made it popular for a wide range of real-world applications.

The degree to which each input contributes to a target class of interest c, or the relevance score of each input concerning c, are among the things we are interested in understanding, given a trained neural network classifier. The fundamental principle behind HADL is to assign a relevance score to each input by

tracking each one's layer-by-layer contribution to the final prediction, $f(x)$. According to the conservation principle, the overall relevance allocated to one layer should match the total relevance allocated to the layer before. This is what the HADL method does. Given two successive layers of a neural network, let's say m and n, the relevance scores meet the following criteria (Eq. [12]):

$$\sum_i R_i^{(m)} = \sum_i R_i^{(n)} = f\left(x\right) \tag{12}$$

In layers m and n, respectively, the relevance scores of individual neurons are denoted by $R_i^{(m)}$ and $R_i^{(n)}$. The rules governing the propagation of relevance scores between two layers by Eq. (9) are varied to accommodate the features of various neural network structures. Eq. (13) illustrates a basic rule:

$$R_i^{(m)} = \sum_j \frac{z_{i,j}}{\sum_k z_{k,j}} R_j^{(n)} \tag{13}$$

where $\sum_k z_{k,j}$ is the total contribution/relevance delivered to neuron j from all linked neurons in layer m before the application of a nonlinear activation function; and $z_{i,j}$ is the contribution/relevance received by neuron j in layer n from an activated neuron i in layer m. This equation demonstrates the conservation principle, which also holds for deactivation, unconnected neurons, and zero weight (Eq. [14]).

$$R_i^{(m)} = \sum_j \frac{z_{i,j}}{\varepsilon + \sum_k z_{k,j}} R_j^{(n)} \tag{14}$$

Despite the HADL rule's many desirable qualities, robustness, and other improvements must be taken into account when applying it to real-world situations (Eq. [15]).

$$R_i^{(m)} = \sum_j \left(\alpha \frac{z_{i,j}^{+}}{\sum_k z_{k,j}^{+}} - \beta \frac{z_{i,j}^{-}}{\sum_k z_{k,j}^{-}} \right) R_j^{(n)} \tag{15}$$

To maintain numerical stability, the denominator has a modest positive term ε in comparison to the fundamental rule, where both the beneficial and detrimental effects from the upper layer n are denoted by $z_{i,j}^{+}$ and $z_{i,j}^{-}$, respectively, and the weights of the positive and negative contributions are controlled by α and β. $\alpha+\beta$ should be in line with the conservation principle. To provide the outcomes with stability and interpretability, the rule prefers the effects of positive contributions over negative ones. One can manually regulate the significance of positive and negative contributions by carefully selecting the values of coefficients α and β.

Temporal convolutional networks with LSTM and other gated neural networks feature a unique calculation called multiplicative interaction in addition to linear mapping computation in multi-layer perceptron architectures. Two neurons are multiplied by one another in this calculation, with one acting as a signal and the other as a gate that regulates the degree to which the signal affects the output (Eq. [16]):

$$a_p = \mathrm{f}(z_g) \cdot g(z_s) \tag{16}$$

where z_g and z_s are two neuron values supplied to the gate and signal unit from earlier layers, respectively, $f(\cdot)$ is the gate unit's activation function, and $g(\cdot)$ is the signal unit's activation function.

In contrast to linear mapping, multiplicative interaction's nonlinearity presents unique challenges related to reassigning importance to the preceding layer. An established redistribution hierarchical method known as "signal-take-all" is used when activation is obtained by multiplying the value of a gate neuron by the value of a signal neuron. This strategy includes (Eq. [17]):

$$(R_g, R_S) = (0, R_{\mathrm{p}}) \tag{17}$$

where the relevance scores for the gate and signal neurons are denoted by R_g and R_s, respectively. To comply with the conservation principle, the gate neuron takes zero, while the signal neuron takes all of the relevant R_p from the top layer.

The HADL is a versatile ML approach that excels in modeling complex data with multiple hierarchical relationships. Its attention mechanisms enhance model interpretability, allowing for a better understanding of the prioritization of features. HADL captures short-term and long-term dependencies, making it ideal for tasks like time-series analysis in WSNs. It also enhances feature learning with its hierarchical structure, allowing for better generalization and robustness in anomaly or fault detection tasks. HADL is adaptable to complex and noisy data, reducing overfitting and improving performance on time-series and sequential data. Its hierarchical nature allows it to scale efficiently to large datasets, making it suitable for real-world applications.

4. Results and Discussion

This section thoroughly analyzes the experimental results to assess the efficacy and efficiency of the suggested approach. The outcomes are compared to several cutting-edge methods using various criteria, including sensitivity, specificity, accuracy, and F1-score. The suggested approach outperformed the other methods by utilizing DL models and sophisticated optimization techniques, attaining near-perfect or perfect values in important measures.

4.1. Experimental Setup

The discussion focuses on interpreting these results, highlighting the impact of the proposed approach on fault detection in WSN, and addressing how the method fills existing research gaps in reliability and precision. The experiments used Python 3.7 as the implementation platform, leveraging libraries such as NumPy, Pandas, TensorFlow/PyTorch, Scikit-learn, and Matplotlib for model development, optimization, and visualization. The system's performance was evaluated under controlled conditions to ensure the robustness and generalizability of the results. The implementation and experimentation were performed on the following system configuration:

(i) Processor: Intel Core i7-12700H (12th Gen) with 14 cores (6 performance + 8 efficiency cores) and a clock speed up to 4.7 GHz
(ii) Random access memory: 16 GB DDR4 3200 MHz, enabling efficient data handling and processing of large datasets
(iii) Storage: 1 TB NVMe SSD, ensuring fast data read/write operations and loading of large models
(iv) Operating system: Windows 11 64-bit, with Python 3.7 as the programming environment
(v) Software frameworks: TensorFlow 2.9, PyTorch 1.12, Scikit-learn 1.1, and Matplotlib 3.5.

This high-performance configuration ensured the smooth execution of computationally intensive tasks, such as hyperparameter tuning, training DL models, and performing iterative optimization. The experiments were iteratively refined to achieve optimal results, balancing computational efficiency and prediction accuracy. The setup included advanced optimization algorithms, fault detection models, and dynamic filtering techniques, tested under controlled conditions to ensure reliable and reproducible results. This environment facilitated seamless experimentation, from pre-processing the WSN-DS to training and evaluating the proposed HADL.

4.2. Dataset Description

The WSN-DS used in this study is a comprehensive and widely used benchmark for fault detection in WSNs. It contains various simulated data representing five distinct classes: normal, grayhole, blackhole, time-division multiple access (TDMA), and flooding. The dataset includes a total of 60,000 instances, with each instance comprising detailed features that capture the behavior and state of network nodes under different conditions. The normal class represents typical, fault-free network operations, while the remaining classes correspond to various network faults and malicious attacks, such as packet-dropping and routing disruptions. Each class is well-balanced, ensuring robust performance evaluation across all fault categories. The dataset provides feature-rich instances, including metrics like node energy levels, packet counts, delays, and routing information, offering a realistic simulation of network scenarios. These features were carefully pre-processed, normalized, and split into training and testing sets to facilitate effective model training and validation. This dataset serves as an ideal foundation for evaluating the performance of fault detection methods in complex WSN environments.

4.3. Performance Metrics

Fig. 3 illustrates the convergence behavior and effectiveness of the RWOA. The initial phase, from the first to the second iterations, shows a significant drop in fitness value, indicating the algorithm's exploratory phase. From the second to the seventh iterations, the plateau phase is marked by a plateau, where the algorithm focuses on refining solutions within a promising region. The further refinement phase decreases slightly to 0.0267, indicating a near-optimal solution and fine-tuning results. The graph demonstrates the algorithm's efficiency in narrowing down the search space, the plateau phase, where the algorithm focuses on exploitation, and the final convergence, where the algorithm has converged to a solution near the global optimum. This graph demonstrates the algorithm's ability to efficiently find an optimal solution while avoiding unnecessary computations beyond the point of diminishing returns.

The confusion matrix represents the performance of a classification model across five classes: Normal, Grayhole, Blackhole, TDMA, and Flooding (Fig. 4). The diagonal elements indicate the correctly classified instances, while off-diagonal elements represent misclassifications. The model performs well overall, with high accuracy for each class, as evidenced by the large diagonal values. For example, normal has 11,857 true positives, with minimal misclassifications.

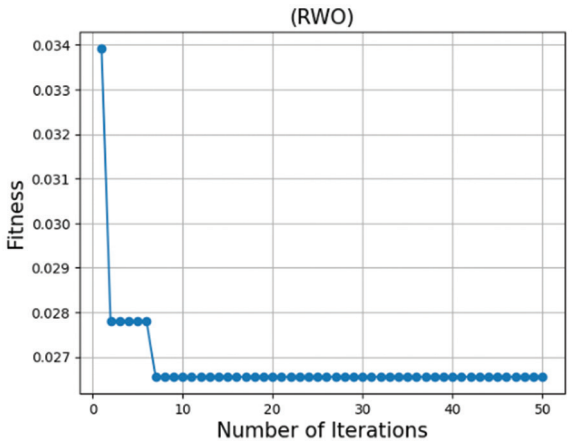

Fig. 3. Convergence behavior and effectiveness of the rank-based whale optimization algorithm

Similarly, grayhole achieves 11,885 correct classifications, though 86 instances were misclassified as blackhole and three as normal. The blackhole class also performed well, with 11,950 true positives and minor misclassifications. For TDMA, the model correctly identified 11,695 instances, though there is some confusion with normal, grayhole, and blackhole. Finally, the Flooding class exhibited near-perfect classification, with 11,991 correct predictions and no significant misclassifications. The confusion matrix highlights the model's robustness but also reveals areas for improvement, such as reducing misclassifications between normal and TDMA and minimizing confusion between grayhole and blackhole.

The AUC values for the five classes (0–4) indicate the model's excellent discriminatory ability across all categories (Fig. 2). AUC values ranged from 0 to 1, with values closer to 1 representing superior performance. Here, the AUC for class 0 (Normal) and class 3 (TDMA) is 0.99, indicating that the model can distinguish these classes from the others with near-perfect accuracy. For class 1 (Grayhole), class 2 (Blackhole), and class 4 (Flooding), the AUC is a perfect 1, demonstrating flawless classification for these classes. This suggests that the model had no false positives or negatives for classes 1, 2, and 4, showing exceptional precision and recall. Overall, the AUC values underscore the model's high reliability and effectiveness in differentiating between all classes, with minimal room for improvement.

The accuracy and loss curves in Fig. 5A and B depict the model's performance during training and testing. In plot 5A, the accuracy curve steadily increases during training, indicating that the model is learning effectively from the data. The testing accuracy also improves and stabilizes, closely aligning with the training accuracy, suggesting good generalization and minimal overfitting. In plot 5B, the loss curve decreases over epochs for both training and testing, reflecting a reduction in prediction errors as the model optimizes its parameters. The convergence of training and testing loss at low values confirms the model's robust learning process. A smooth and stable trajectory for both accuracy and loss curves indicates that the model training is well-tuned, with no signs of underfitting or overfitting, and performs consistently on unseen test data.

Fig. 6 shows the performance metrics for training and testing. With an overall accuracy of 99%, the model performed exceptionally well in the testing phase across all five classes of WSNs. With values near 0.99 or 1.00, the model's accuracy, recall, and F1-scores were continuously high, demonstrating its capacity to accurately detect occurrences of each class while reducing false positives and false negatives. With a perfect score, the flooding class exhibited faultless detection. The performance of other classes, such as blackhole and grayhole, was also strong. Weighted and macro average measures further support the model's balanced performance across classes. The model's outstanding performance during training and testing, together with its ability to balance accuracy, recall, and F1-score, shows its usefulness in real-world situations where reliable and precise fault classification is required.

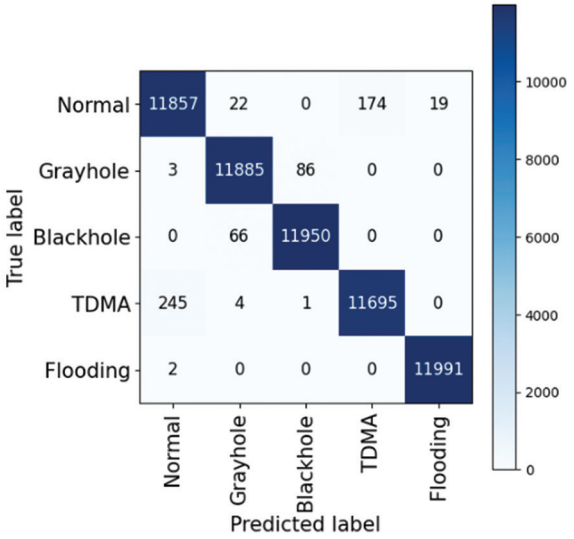

Fig. 4. Confusion matrix
Abbreviation: TDMA: Time-division multiple access

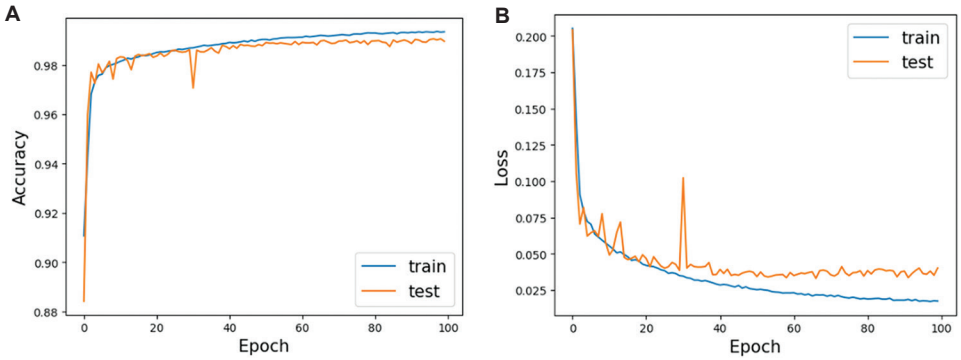

Fig. 5. (A and B) Accuracy and loss curves

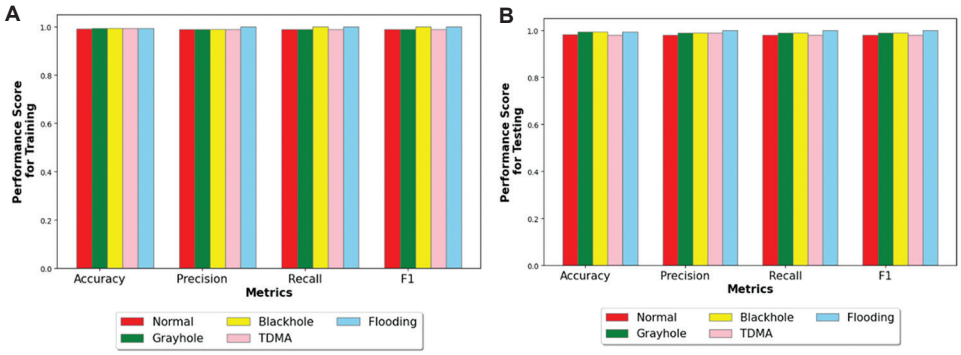

Fig. 6. (A and B) Performance metrics for training and testing
Abbreviation: TDMA: Time-division multiple access

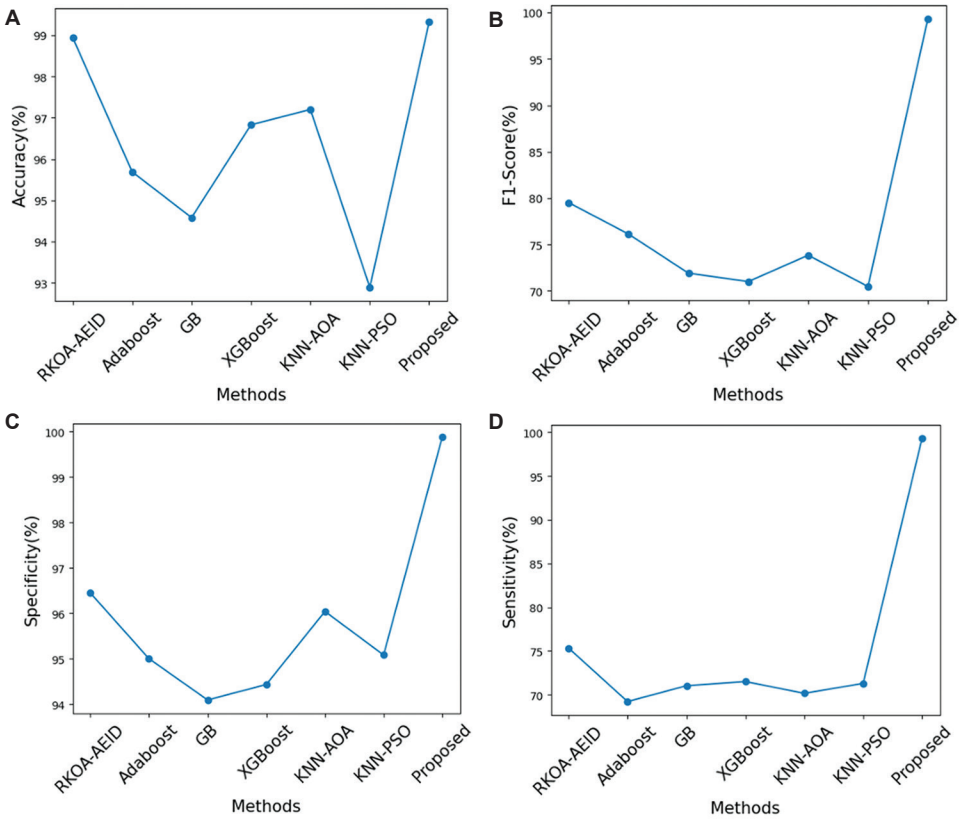

Fig. 7. (A-D) Comparative analysis for performance metrics
Abbreviations: AOA: Angle or arrival; GB: Gradient boosting; KNN: K-nearest neighbor; RKOA-AEID: Red kite optimization algorithm-average ensemble model for intrusion detection; PSO: Particle swarm optimization; XG Boost: Extreme gradient boosting

4.4. Comparison Metrics

Fig. 7 shows the performance of various methods, including RKOA-AEID (Alruwaili et al., 2023), Adaboost (Aljebreen et al., 2023), gradient boosting (Aljebreen et al., 2023), extreme gradient boosting (Alqahtani et al., 2019), KNN-angle of arrival (Liu et al., 2022), KNN-particle swarm optimization (Liu et al., 2022), and the proposed method, across four evaluation metrics: accuracy, F1-score, specificity, and sensitivity (Table 1). The proposed method achieved the highest accuracy (99%), demonstrating superior reliability in fault detection. RKOA-AEID performed well (98%), while Adaboost and KNN-particle swarm optimization performed moderately (94%). Gradient boosting, extreme gradient boosting, and KNN-angle of arrival exhibited intermediate results (97%). The proposed method excelled with a perfect F1-score (100%), indicating an exceptional balance between precision and recall. Adaboost and gradient boosting

Table 1. Comparative chart of the proposed model with conventional methods

Methods	Accuracy	Sensitivity	Specificity	F-score
Red kite optimization algorithm-average ensemble model for intrusion detection (Alruwaili et al., 2023)	98.94	75.33	96.45	79.52
AdaBoost (Aljebreen et al., 2023)	95.69	69.22	95.00	76.13
Gradient booting (Aljebreen et al., 2023)	94.58	71.03	94.09	71.92
Extreme gradient boosting (Alqahtani et al., 2019)	96.83	71.51	94.43	71.01
K-nearest neighbor-angle of arrival (Liu et al., 2022)	97.20	70.16	96.04	73.85
K-nearest neighbor-particle swarm optimization (Liu et al., 2022)	92.89	71.30	95.08	70.48
Proposed	99.25	98.74	99.32	98.39

lagged (75%), reflecting weaker handling of false positives or false negatives. Extreme gradient boosting and KNN-angle of arrival performed moderately (97%). The proposed method consistently outperforms all other techniques, achieving perfect F1, specificity, and sensitivity scores and near-perfect accuracy.

5. Conclusion

This research presents a unified framework for fault detection in WSNs that effectively combines advanced noise filtering, optimized feature selection, and a sophisticated DL architecture. The proposed approach leverages a DNF technique with adaptive thresholding to cleanse the data while preserving its critical aspects, employs the RWOA to select the most relevant features, and utilizes an HADL model to capture both short-term and long-term dependencies in sensor data. Experimental evaluations of the WSN-DS confirm the framework's exceptional performance, achieving an accuracy of 99.25%, sensitivity of 98.74%, specificity of 99.32%, and an F-score of 98.39%. These results highlight the framework's capacity to reliably detect faults and reduce false alarms, ultimately enhancing network reliability and extending the operational lifespan of WSNs. The integration of these advanced methodologies not only addresses existing challenges in fault detection but also establishes a robust foundation for future enhancements, including real-time deployment and the incorporation of multi-modal data.

Funding

None.

Conflict of Interest

The authors declare that they have no conflict of interest.

Availability of Data

Not applicable.

References

Alghamdi, R., & Bellaiche, M. (2023). A cascaded federated deep learning based framework for detecting wormhole attacks in IoT networks. *Computers and Security Journal*, 125, 103014. https://doi.org/10.1016/j.cose.2022.103014

Aljebreen, M., Alohali, M.A., Saeed, M.K., Mohsen, H., Al Duhayyim, M., Abdelmageed, A.A., et al. (2023). Binary chimp optimization algorithm with ML based intrusion detection for secure IoT-assisted wireless sensor networks. *Sensors*, 23(8), 4073. https://doi.org/10.3390/s23084073

Almomani, O. (2021). A hybrid model using bio-inspired metaheuristic algorithms for network intrusion detection system. *Computers, Materials and Continua*, 68, 409–429. https://doi.org/10.32604/cmc.2021.016113

Alqahtani, M., Gumaei, A., Mathkour, H., & Maher Ben Ismail, M. (2019). A genetic-based extreme gradient boosting model for detecting intrusions in wireless sensor networks. *Sensors (Basel)*, 19(20), 4383. https://doi.org/10.3390/s19204383

Alruhaily, N.M., & Ibrahim, D.M. (2021), A multi-layer machine learningbased intrusion detection system for wireless sensor networks. *The International Journal of Advanced Science and Computer Applications*, 12(4), 281–288. https://doi.org/10.14569/ijacsa.2021.0120437

Alruwaili, F.F., Asiri, M.M., Alrayes, F.S., Aljameel, S.S., Salama, A.S., & Hilal, A.M. (2023). Red kite optimization algorithm with average ensemble model for intrusion detection for secure IoT. *IEEE Access*, 11, 131749–131758. https://doi.org/10.1109/access.2023.3335124

Chandre, P., Mahalle, P., & Shinde, G. (2022). Intrusion prevention system using convolutional

neural network for wireless sensor network. *International Journal of Artificial Intelligence*, 2252(8938), 8938. https://doi.org/10.11591/ijai.v11.i2.pp504-515

Chataut, R., Phoummalayvane, A., & Akl, R. (2023). Unleashing the power of IoT: A comprehensive review of IoT applications and future prospects in healthcare agriculture, smart homes, smart cities, and industry 4.0. *Sensors (Basel)*, 23(16), 7194. https://doi.org/10.3390/s23167194

Elsaid, S.A., & Albatati, N.S. (2020). An optimized collaborative intrusion detection system for wireless sensor networks. *Soft Computing*, 24(16), 12553–12567. https://doi.org/10.1007/s00500-020-04695-0

Gebremariam, G.G., Panda, J., & Indu, S. (2023). Design of advanced intrusion detection systems based on hybrid machine learning techniques in hierarchically wireless sensor networks. *Connectection Science*, 35(1), 2246703. https://doi.org/10.1080/09540091.2023.2246703

Ghazal, T. (2022). Data fusion-based machine learning architecture for intrusion detection. *Computers, Materials and Continua*, 70(2), 3399–3413. https://doi.org/10.32604/cmc.2022.020173

Heidari, A., & Jabraeil Jamali, M.A. (2022). Internet of things intrusion detection systems: A comprehensive review and future directions. *Cluster Computing*, 26, 1–28. https://doi.org/10.1007/s10586-022-03776-z

Liu, G., Zhao, H., Fan, F., Liu, G., Xu, Q., & Nazir, S. (2022). An enhanced intrusion detection model based on improved kNN in WSNs. *Sensors (Basel)*, 22(4), 1407. https://doi.org/10.3390/s22041407

Meng, D., Dai, H., Sun, Q., Xu, Y., & Shi, T. (2022). Novel wireless sensor network intrusion detection method based on lightGBM model. *IJAM - IAENG International Journal of Applied Mathematics*, 52(4), 23.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, 95, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Nimbalkar, A..D., Azmat, A., & Patil, Y. (2023). Security issues in wireless sensor networks. *I-Manager's Journal on Wireless Communication Networks*, 11(2), 32. https://doi.org/10.26634/jwcn.11.2.19780

Pandey, J.K., Kumar, S., Lamin, M., Gupta, S., Dubey, R.K., & Sammy, F. (2022). A metaheuristic autoencoder deep learning model for intrusion detector system. *Mathematical Problems in Engineerin*g, 2022, 3859155. https://doi.org/10.1155/2022/3859155

Putrada, A.G., Alamsyah, N., Pane, S.F., & Fauzan, M.N. (2022). Xgboost for Ids on WSN Cyber Attacks with Imbalanced Data. In: *2022 International Symposium on Electronics and Smart Devices (ISESD)*, p1–7.

Qaiwmchi, N.A.H., Amintoosi, H., & Mohajerzadeh, A. (2020). Intrusion detection system based on gradient-corrected online sequential extreme learning machine. *IEEE Access*, 9, 4983–4999. https://doi.org/10.1109/ACCESS.2020.3047933

Ravindra, C., Kounte, M.R., Lakshmaiah, G.S., & Prasad, V.N. (2023). Etelmad: Anomaly detection using enhanced transient extreme machine learning system in wireless sensor networks. *Wireless Personal Communications*, 130(1), 21–41. https://doi.org/10.1007/s11277-023-10271-0

Rezvi, M.A., Moontaha, S., Trisha, K.A., Cynthia, S.T., & Ripon, S. (2021). Data mining approach to analyzing intrusion detection of the wireless sensor network. *Indonesian Journal of Electrical Engineering and Computer Science*, 21(1), 516–523. https://doi.org/10.11591/ijeecs.v21.i1.pp516-523

Sezgin, A., & Boyacı, A. (2023). Aid4i: An intrusion detection framework for industrial internet of things using automated machine learning. *Computers, Materials and Continua*, 76(2), 40287.

Sharmin, S., Ahmedy, I., & Md Noor, R. (2023). An energy-efficient data aggregation clustering algorithm for wireless sensor networks using hybrid PSO. *Energies*, 16(5), 2487. https://doi.org/10.3390/en16052487

Singh, N., Virmani, D., & Gao, X.Z. (2020). A fuzzy logic-based method to avert intrusions in wireless sensor networks using WSN-DS dataset. *International Journal of Computer Applications*, 19(3), 2050018. https://doi.org/10.1142/S1469026820500182

Talukder, M.A., Islam, M.M., Uddin, M.A., Akhter, A., Hasan, K.F., & Moni, M.A. (2022). Machine learning-based lung and colon cancer detection using deep feature extraction and ensemble learning, *Expert Systems with Applications*, 205, 117695. https://doi.org/10.1016/j.eswa.2022.117695

Talukder, M.A., Hasan, K.F., Islam, M.M., Uddin, M.A., Akhter, A., Yousuf, M.A., et al. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, 72, 103405. https://doi.org/10.48550/arXiv.2212.04546

Talukder, M.A., Islam, M.M., Uddin, M.A., Hasan, K.F., Sharmin, S., Alyami, S.A., et al. (2024). Machine

learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, 11, 33.

Tan, X., Su, S., Huang, Z., Guo, X., Zuo, Z., Sun, X., et al. (2019). Wireless sensor networks intrusion detection based on smote and the random forest algorithm. *Sensors (Basel)*, 19(1), 203. https://doi.org/10.3390/s19010203

Yakubu, M.M., & Maiwada, U.D. (2023). Resource limitations for wireless sensor networks to establish a comprehensive security system in the 5g network. *Umyu Scientifica*, 2(2), 44–52. https://doi.org/10.1007/s10207-024-00833-z

AUTHOR BIOGRAPHIES

R. Gayathri is a student in the Department of Computer Science and Engineering at Siddaganga Institute of Technology, affiliated with Visvesvaraya Technological University, Karnataka, India. Her areas of interest include machine learning, artificial intelligence, and data analytics.

K. N. Shreenath is an Associate Professor in the Department of Computer Science and Engineering at Siddaganga Institute of Technology, affiliated with Visvesvaraya Technological University, Karnataka, India. He has several years of teaching and research experience. His research interests include computer networks, data mining, artificial intelligence, and software engineering. He has guided numerous student projects and published papers in national and international journals and conferences.