A Hybrid Approach for Intrusion Detection in the Internet of Things Using Harris Hawks Optimization and Deep Learning Algorithms

Reza Kohan[1], Hamid Barati[2], Ali Barati[3]

1- Institute of Artificial Intelligence and Social and Advanced Technologies, Dez.C., Islamic Azad University, Dezful, Iran.
Email: rezakohan.rk061@gmail.com
2- Institute of Artificial Intelligence and Social and Advanced Technologies, Dez.C., Islamic Azad University, Dezful, Iran.
Email: Hamid.barati@iau.ac.ir (Corresponding author)
3- Institute of Artificial Intelligence and Social and Advanced Technologies, Dez.C., Islamic Azad University, Dezful, Iran.
Email: alibarati@iau.ac.ir

ABSTRACT:
Intrusion detection in Internet of Things (IoT)-based smart cities is essential due to the increasing volume and complexity of cyberattacks. Traditional detection systems face two major challenges: achieving high accuracy and minimizing false alarms, particularly in large-scale and heterogeneous IoT networks. This paper proposes a novel hybrid intrusion detection system that combines Harris Hawks Optimization (HHO) for feature selection with a multi-layer neural network enhanced by learning automata for adaptive classification. The HHO algorithm efficiently reduces input dimensionality by selecting the most relevant features, while the learning automata optimize the network's weights dynamically, improving training stability and robustness. The proposed system is evaluated using the KDDCup99 dataset under both binary and multiclass scenarios. Experimental results show an average accuracy of 96.53%, a true positive rate (TPR) of 94.91%, and a false positive rate (FPR) of 2.80%. Compared to recent baseline models, the proposed method demonstrates superior performance in accuracy and false alarm reduction, confirming its suitability for real-time intrusion detection in dynamic IoT-based environments.

KEYWORDS: Intrusion Detection, Internet of Things, Harris Hawks Optimization, Neural Network, Learning Automata.

1. INTRODUCTION

With the rapid advancement of smart city technologies and the widespread deployment of Internet of Things (IoT) devices, urban environments are becoming increasingly interconnected, automated, and data-driven. These systems rely on distributed sensor networks, cloud-based platforms, and intelligent decision-making to improve urban services such as traffic control, energy management, public safety, and environmental monitoring [1]. However, the growing complexity and openness of IoT-based smart city infrastructures have also made them highly susceptible to cyberattacks, posing significant threats to data confidentiality, system availability, and operational integrity [2].

Intrusion Detection Systems (IDS) play a crucial role in safeguarding smart city networks by monitoring network traffic and detecting anomalous or malicious behavior [3]. Traditional IDS approaches, including signature-based and rule-based models, often fall short in detecting novel or evolving attacks and struggle to adapt to the dynamic and large-scale nature of IoT networks [4]. Moreover, the high volume of data and the heterogeneity of devices introduce significant challenges in scalability, accuracy, and false alarm reduction [5].

To address these challenges, machine learning (ML) and metaheuristic optimization algorithms have gained attention for their ability to model complex attack patterns and improve detection performance [6]. In particular, feature selection

Paper type: Research paper
https://doi.org/xxx
Received: 18 January 2025, Revised: 15 February 2025, Accepted: 3 May 2025, Published: 1 June 2025
How to cite this paper: R. Kohan, H. Barati, A. Barati, **"A Hybrid Approach for Intrusion Detection in the Internet of Things Using Harris Hawks Optimization and Deep Learning Algorithms"**, *Majlesi Journal of Telecommunication Devices*, Vol. 14, No. 2, pp. 89-104, 2025.

has emerged as a vital preprocessing step to reduce redundant information, improve classifier efficiency, and mitigate the impact of noisy or irrelevant data [7]. Similarly, the quality of the classifier, including its training stability and adaptability, plays a key role in determining the effectiveness of the IDS [8].

We propose a novel hybrid intrusion detection method that integrates Harris Hawks Optimization (HHO) for optimal feature selection and a multi-layer neural network optimized using learning automata for adaptive classification. The HHO algorithm, inspired by the cooperative hunting behavior of Harris hawks, efficiently explores the feature space to identify the most informative attributes. Meanwhile, the learning automata dynamically adjust the weights of the neural network based on training feedback, enhancing learning stability and classification accuracy.

The contributions of this work are summarized as follows:

- We design a feature selection mechanism based on Harris Hawks Optimization to reduce computational complexity and improve detection performance.
- We introduce a learning automata-driven neural network training approach to enhance adaptability and avoid overfitting.
- We validate the proposed system using the KDDcup99 benchmark dataset and demonstrate superior performance in terms of accuracy, false positive rate, and robustness across different attack types.
- We compare our model with two recent methods and show its effectiveness in handling class imbalance and rare attack detection.

The remainder of the paper is organized as follows: Section 2 reviews related work on intrusion detection in IoT environments. Section 3 presents the problem formulation and background concepts. Section 4 details the proposed methodology. Section 5 discusses the experimental results and comparative analysis. Finally, Section 6 concludes the paper and outlines directions for future work.

2. RELATED WORK

In recent years, intrusion detection in the Internet of Things (IoT) has attracted growing attention due to the proliferation of connected devices and the increasing sophistication of cyber threats. A variety of intelligent and hybrid models have been proposed to address the unique constraints of IoT environments, including resource limitations, data heterogeneity, and real-time processing requirements. Khan et al. [9] developed a deep neural network (DNN)-based IDS tailored for MQTT-based IoT environments. Their model leveraged three levels of feature abstraction—packet-flow, uni-flow, and bi-flow—achieving over 99% accuracy in uni-flow and bi-flow detection. However, classification accuracy declined for packet-flow due to class imbalance.

Zhao et al. [10] proposed a lightweight DNN with PCA-based feature reduction for low-resource IoT devices. Their architecture integrated inverted residuals and channel-wise fusion mechanisms to optimize performance. While the model maintained high accuracy and efficiency, its performance varied across feature types, particularly for packet-level data.

Wahab [11] introduced an online deep learning approach for intrusion detection, which addressed concept drift via Hedge Backpropagation. This two-phase method—detecting changes and adapting the model—enabled real-time learning and outlier detection, demonstrating strong resistance to evolving attack patterns.

Vishwakarma and Kesswani [12] presented the DIDS system based on DNN, designed for real-time detection using Netflow-based data. Their approach combined PCA for dimensionality reduction, dropout layers to avoid overfitting, and Hedge Backpropagation for adaptability. The model achieved high classification accuracy in both binary and multiclass tasks.

Chatterjee and Hanawal [13] developed a hybrid ensemble-based IDS called PHEC, combining KNN and Random Forest to improve detection accuracy while reducing false positives. They extended this model to a federated learning setting to preserve data privacy across distributed IoT devices. A noise-tolerant version of PHEC was also introduced to address label noise using a weighted convex loss. Evaluations on four benchmark datasets showed high TPR and low FPR in both clean and noisy scenarios, with federated performance closely matching the centralized setup.

Ngo et al. [14] designed the HH-NIDS framework using lightweight neural networks implemented on heterogeneous hardware platforms, including MAX78000EVKIT and FPGA. Their system achieved over 99% accuracy on UNSW-NB15 and IoT-23 datasets while significantly reducing energy consumption and inference time.

Awajan [15] proposed a dynamic IDS using deep neural networks equipped with adaptive feature extraction modules. Their model achieved a 93.21% average detection rate across five common attack types. Despite high performance, the system's complexity posed challenges for real-time deployment.

Wang et al. [16] introduced BT-TPF, a knowledge-distillation-based framework using a large Vision Transformer (teacher) and a compact Poolformer (student) for lightweight IDS. Their model achieved over 99% accuracy on CIC-IDS2017 and TON_IoT datasets, though training complexity and reliance on a powerful teacher model remained concerns.

Qaddos et al. [17] proposed a similar lightweight detection system by combining Siamese networks and knowledge distillation, also achieving over 99% accuracy. The model was optimized for limited-resource environments but required precise calibration and rich training data to maintain robustness.

Wang et al. [18] presented FeCo, a federated contrastive learning framework designed to preserve data privacy while improving detection performance. FeCo effectively handled non-IID data and achieved up to 8% accuracy improvement over state-of-the-art models, though computational complexity in large-scale deployments remained a limitation.

Lin et al. [19] proposed E-GRACL, a GNN-based IDS that modeled traffic correlations via graph structures. By enhancing the GraphSAGE model with attention and gating mechanisms, and integrating contrastive learning, E-GRACL demonstrated high accuracy in both binary and multiclass settings, but at the cost of computational overhead and complex model tuning.

Table 1 provides a comparative overview of recent IDS approaches in IoT contexts, summarizing their core techniques, advantages, and known limitations.

Table 1. Comparative summary of recent intrusion detection approaches in IoT environments

Ref	Year	Method / Model	Dataset(s) Used	Key Techniques	Strengths	Limitations
[9]	2021	DNN for MQTT IDS	MQTT-IoT-IDS2020	Multi-level flow (Uni/Bi/Packet), Adam optimizer	High accuracy, flow-specific feature modeling	Poor performance on Packet-flow due to class imbalance
[10]	2021	Lightweight DNN	Real-world datasets	PCA + DNN + NID loss	Compact model for low-resource devices	Lower precision on complex attack types
[11]	2022	Online DL + Hedge Backprop	Simulated stream data	Concept drift handling, PCA, outlier detection	Fast adaptation, reduced training-test gap	Complex update mechanism
[12]	2022	DIDS – Real-time DNN	NetFlow-based IoT traffic	PCA + Dropout + Hedge Backprop	Real-time detection, adaptive learning	May require high processing capacity
[13]	2022	PHEC + Federated Learning	NSL-KDD, DS2OS, Gas Pipeline, Water Tank	KNN + RF ensemble, Fed-stacking, weighted loss	Privacy-preserving, noise-tolerant	Higher computation, edge-device constraints
[14]	2023	HH-NIDS (Heterogeneous HW)	UNSW-NB15, IoT-23	Lightweight NN + FPGA & microcontroller	Energy-efficient, real-time inference	Needs specialized hardware
[15]	2023	Adaptive DNN	Custom dataset	Dynamic feature extraction, attack reduction phase	Modular, flexible model	High model complexity, slow inference
[16]	2024	BT-TPF (Knowledge Distill.)	CIC-IDS2017, TON_IoT	Vision Transformer + Poolformer	Model compression, high accuracy	Training dependent on teacher model
[17]	2024	Enhanced BT-TPF	CIC-IDS2017, TON_IoT	Siamese Net + Knowledge distillation	Efficient for constrained IoT	Sensitivity to data variation
[18]	2025	FeCo (Federated + Contrastive)	NSL-KDD, BaIoT	Fed. learning + Feature filtering + Contrastive learning	Privacy-preserving, handles non-IID	Slower convergence, complex coordination

[19]	2025	E-GRACL (Graph-based IDS)	Three benchmark sets	GNN (GraphSAGE+) + Global attention + Contrastive learning	Captures topology, high detection in binary/multiclass	High computational cost, tuning required

3. PROBLEM FORMULATION AND BACKGROUND CONCEPTS

3.1. Problem Definition

Intrusion detection in IoT-enabled smart city environments presents a challenging task due to the high volume of data, the heterogeneous nature of connected devices, and the presence of both known and unknown cyber threats. The goal is to design a detection system capable of identifying a wide variety of attacks—including volumetric, probing, and stealthy intrusions—while maintaining high accuracy and low false alarm rates [20].

Formally, let $D = \{x_1, x_2, \dots, x_n\}$ be a dataset of network traffic records, where each instance $x_i \in \mathbb{R}^m$ is characterized by m features. The task is to learn a function $f: \mathbb{R}^m \to C$, where C= {normal, DoS, Probe, R2L, U2R}, that maps each input instance to its correct class label. Due to high dimensionality and data imbalance, the learning function must be trained on a reduced and optimized feature subset $F' \subseteq F$, where $|F'| \ll |F|$, without sacrificing classification performance.

This problem requires a two-step solution:

1. Optimal Feature Selection: Identify a compact subset of relevant features to improve classification accuracy and reduce computational cost.
2. Robust Classification: Train an adaptable and generalizable model capable of handling diverse and imbalanced data efficiently.

3.2. Background Concepts

3.2.1 Feature Selection in Intrusion Detection

Feature selection plays a critical role in reducing noise and irrelevant information in high-dimensional data. It aims to select a subset of the most discriminative features that contribute significantly to the learning process. This not only improves the detection accuracy but also speeds up the training phase and prevents over fitting [21]. In intrusion detection, feature selection is particularly important because many features in datasets like KDDcup99 [22] are redundant or weakly correlated with attack behavior.

3.2.2. Harris Hawks Optimization (HHO)

Harris Hawks Optimization (HHO) [23] is a recent nature-inspired metaheuristic algorithm that simulates the cooperative hunting strategy of Harris hawks in nature. It employs both exploration (global search) and exploitation (local search) mechanisms, adapting its strategy based on the prey's escaping energy and the current fitness of candidate solutions. In the context of feature selection, each hawk represents a binary vector corresponding to a potential subset of features, and the fitness function evaluates classification performance using this subset. HHO has been shown to converge rapidly and explore diverse regions of the solution space, making it suitable for dynamic IDS environments.

3.2.3. Neural Networks for Classification

Artificial Neural Networks (ANNs) [24] are widely used in classification tasks due to their strong learning capabilities and non-linear mapping functions. In this work, a multi-layer feedforward neural network is employed as the core classifier. It is composed of an input layer, one or more hidden layers, and an output layer, and is trained using supervised learning with backpropagation. However, traditional training may suffer from local minima and slow convergence, especially when using large or imbalanced datasets.

3.2.4. Learning Automata for Adaptive Training

To overcome the limitations of standard backpropagation, Learning Automata (LA) [25] are integrated into the training process. A learning automaton is a probabilistic decision-making mechanism that adjusts its actions based on feedback from the environment. In this context, the automaton evaluates training progress by monitoring changes in classification error. It rewards configurations that reduce error and penalizes those that degrade performance. This adaptive process helps in avoiding local minima and improves generalization in neural networks, particularly under non-stationary or noisy training conditions.

4. PROPOSED METHOD

This section describes the proposed intrusion detection method for smart city-based Internet of Things (IoT) environments. The approach combines advanced feature selection through Harris Hawks Optimization (HHO) with an enhanced multi-layer neural network (MLNN) supported by learning automata for adaptive weight tuning. The method follows a structured pipeline including data preprocessing, feature selection, classification, and training optimization. A complete overview of the proposed architecture is illustrated in Fig. 1.

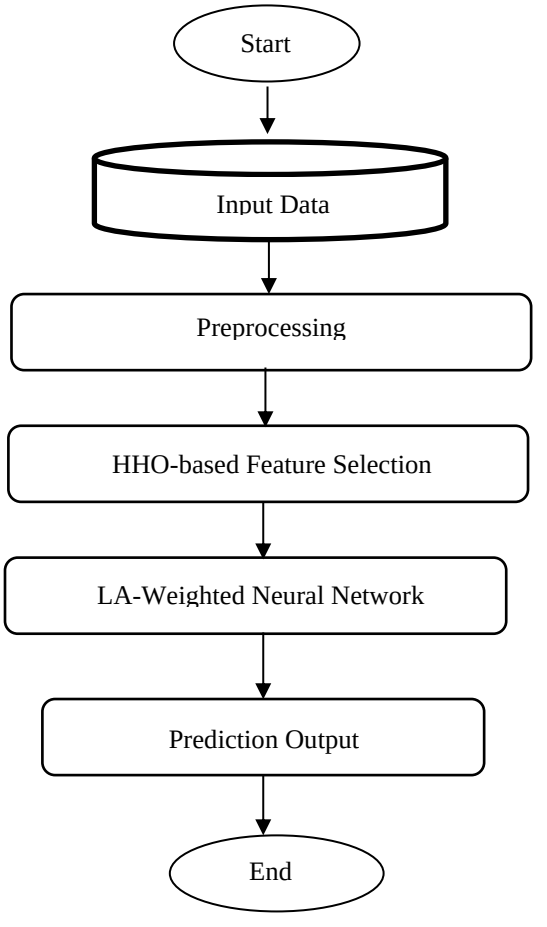

Fig. 1. Architecture of the proposed hybrid intrusion detection system.

4.1. Overview of the Architecture

The overall architecture of the proposed system includes the following major phases:

1. Input Dataset: The system is trained and evaluated using the KDDcup99 dataset.
2. Preprocessing: Includes normalization, categorical encoding, and dimensional transformation.
3. Feature Selection: Utilizes Harris Hawks Optimization (HHO) to select optimal subsets of features.
4. Classification: A supervised multi-layer neural network enhanced by learning automata.
5. Intrusion Detection: The final model classifies traffic as normal or malicious (DoS, Probe, R2L, U2R).

The goal is to improve accuracy, reduce computational complexity, and minimize false alarms in detecting network intrusions within smart city infrastructures.

4.1. Dataset Description and Preprocessing

The KDDcup99 dataset, derived from real network traffic simulations, contains 41 features per connection and labels for different attack types. This dataset is widely used for intrusion detection research, but is known to contain noise and class imbalance.

4.2.1. Label Consolidation

For simplicity and effectiveness, records are grouped into five categories:

- Normal
- Denial of Service (DoS)
- Probe
- Remote to Local (R2L)
- User to Root (U2R)

To mitigate severe class imbalance inherent in the original KDDCup99 dataset, oversampling techniques were applied to minority classes (R2L and U2R), leading to improved representativeness in test subsets.

4.2.2. Categorical Feature Encoding

Three categorical features (protocol_type, service, flag) are converted to numerical form using one-hot encoding:

- protocol_type: 3 values (TCP, UDP, ICMP)
- service: 70 distinct values
- flag: 11 values

Resulting in 122 features after transformation.

4.2.3. Normalization

Features with highly skewed distributions (e.g., src_bytes, dst_bytes, duration) are normalized to the range [0,1] using min-max scaling (Eq. 1):

$$x_i = \frac{x_i - Min}{Max - Min} \quad (1)$$

x_i is the i$th component of the feature vector. **Min** and **Max** refer to the minimum and maximum values of the ith column in the dataset, respectively. Normalization improves neural network performance by preventing dominance of large-scale features.

4.2. Feature Selection using Harris Hawks Optimization (HHO)

To reduce dimensionality and enhance classifier performance, Harris Hawks Optimization (HHO) is employed. HHO simulates the hunting behavior of Harris hawks with a balance between exploration and exploitation phases.

4.3.1. Initialization

Each hawk represents a binary vector indicating whether a feature is selected (1) or not (0). Initial parameters are:

- Population size: 20 hawks
- Maximum iterations: 100
- Feature encoding: 41-dimensional binary vectors (post one-hot expansion to 122 dimensions)

The Harris Hawks Optimization (HHO) algorithm is a continuous algorithm and generates continuous solutions. However, the proposed method for feature selection requires discrete results. Therefore, the solution values generated by HHO must be converted into binary values. To achieve this, the results are rounded to 0 or 1. An example of the initial population is shown in Table. 2.

Table 2. An example of the initial population

	Feature 1	Feature 2	Feature 3	Feature 4	.	.	.	Feature 39	Feature 40	Feature 41
Solution 1	1	0	0	1	.	.	.	1	1	1
Solution 2	0	0	1	1	.	.	.	0	1	0
Solution 3	1	0	1	0	.	.	.	0	0	0
	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .
Solution n-1	1	1	0	0	.	.	.	1	1	0
Solution n	1	1	0	0	.	.	.	1	1	0

4.3.2. Fitness Function

At this stage, the evaluation of candidate solutions is performed. After generating the initial population, the performance of each solution is assessed using a fitness function within the Harris Hawks Optimization (HHO) algorithm. The fitness of each hawk represents the quality of the feature subset it has selected. The primary objective is to identify optimal or near-optimal parameters that yield the most accurate solution.

The fitness function in the proposed method aims to minimize the classification error while maximizing the true positive rate and selecting the most informative feature subset. To achieve this, each candidate feature subset is evaluated based on three criteria: true positive rate (TPR), error rate, and the number of selected features. The fitness function used in the proposed method is defined by Eq. (2).

$$Fitness = R_{tp} + (1 - R_E) + (1 - \frac{N_F}{N}) \quad (2)$$

R_{tp} represents the true positive rate, calculated using Eq. (3); R_E denotes the error rate, computed based on Eq. (4); and N_F is the number of selected features, while N is the total number of features.

According to Equation (2), the fitness value increases when the true positive rate is higher, the error rate is lower, and the selected feature subset is smaller. This ensures that the proposed method not only improves classification performance but also promotes dimensionality reduction by selecting the most relevant features.

This formulation ensures the selected feature subset achieves high classification accuracy with minimal dimensionality.

The true positive rate (TPR) is calculated by dividing the number of true positives (TP) by the sum of true positives (TP) and false negatives (FN). This metric reflects the model's ability to correctly identify positive instances. Eq. (3) presents the formula used to compute the true positive rate.

$$R_{tp} = \frac{TP}{TP+FN} \quad (3)$$

TP denotes the number of true positive instances, and FN represents the number of false negatives.

The error rate refers to the proportion of instances that are incorrectly predicted by the classifier out of all samples. It reflects the overall misclassification and is calculated using Eq. (4).

$$R_E = \frac{FP + FN}{TP + FN + TN + FP} \quad (4)$$

Where TP denotes the number of true positives, FP represents the number of false positives, TN is the number of true negatives, and FN refers to the number of false negatives. These values form the foundation of the confusion matrix and are essential for evaluating the performance of the classification model.

4.3.3. Exploration and Exploitation Phases

- Exploration: Hawks perform random search via Lévy flight and stochastic positioning.
- Exploitation: Hawks adjust their positions based on the best solution (prey).

The transition between phases depends on the prey's escaping energy E, updated each iteration (Eq. 5):

$$E = 2E_0(1 - \frac{t}{T}) \quad (5)$$

E represents the escaping energy of the prey, T denotes the maximum number of iterations, and E_0 is the initial energy of the prey. The value of E_0 can vary randomly within the range (–1, +1) in each iteration. When E_0 decreases from 0 to –1, it implies that the prey (rabbit) is becoming physically weaker. Conversely, when E_0 increases from 0 to +1, it indicates that the prey is gaining strength.

The exploration phase in the Harris Hawks Optimization (HHO) algorithm involves the hawks moving randomly within the search space to discover better positions. This process enables the hawks to explore diverse regions and prevents stagnation in a specific area. The movement of hawks in this phase is random and guided by techniques such as Levy Flight, a stochastic movement pattern characterized by a combination of short and long jumps. This strategy allows the hawks to explore distant regions through long jumps and examine local areas through short jumps.

In the HHO algorithm, the Harris hawks represent candidate solutions, and the best candidate at each iteration is

considered the target prey or the near-optimal solution. Initially, hawks perch randomly in the environment, waiting for opportunities. Depending on a probability parameter q, two different strategies for locating the prey are applied:

- If q<0.5, the hawks perch based on the positions of other hawks and the rabbit, simulating coordinated waiting behavior.
- If q≥0.5, the hawks perch randomly on tall trees (i.e., random positions within the population's range).

This behavior is mathematically described by Eq. (6):

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t) & q \geq 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - r_3\left(LB + r_4(UB - LB)\right) & q < 0.5 \end{cases} \quad (6)$$

Where $X(t+1)$ is the position vector of a hawk in iteration t+1, X(t) is its current position, $X_{rand}(t)$ is a randomly selected hawk's position, $X_{rabbit}(t)$ is the position of the rabbit (best solution), $X_m(t)$ is the mean position of the current population, r_1, r_2, r_3, r_4 and q are random numbers in the range (0,1), updated at each iteration, LB and UB denote the lower and upper bounds of the search space.

In the first rule (q≥0.5), new solutions are generated based on a random position influenced by both the previous position and the location of other hawks. In the second rule (q<0.5), the update rule incorporates a random location within the search boundaries, along with the difference between the current best position and the population mean. The variables r_3 and r_4 act as scaling factors to enhance the stochastic behavior of the hawks within the defined bounds.

4.3.4. Stopping Criteria

HHO terminates when either:

- The best fitness value does not improve significantly (≤ 0.001), or
- The algorithm reaches the maximum number of iterations.

The best feature subset obtained is passed to the classifier. On average, HHO reduced the number of input features from 122 to approximately 38, representing a 69% reduction in dimensionality without compromising classification performance.

4.3. Intrusion Classification using Neural Network with Learning Automata

The selected features are used to train a supervised multi-layer feedforward neural network. To enhance convergence, training is supported by a Learning Automaton (LA) which dynamically adjusts weight updates.

4.4.1. Neural Network Architecture

The classifier consists of:

- Input Layer: Corresponding to the number of selected features
- Hidden Layer: Nonlinear transformation using sigmoid activation
- Output Layer: Binary output (normal vs. attack) or multiclass if needed

Weight initialization is random, and the model is trained using backpropagation with error feedback. The training process is primarily based on standard backpropagation. However, to improve convergence and generalization, learning automata are integrated into the training loop, dynamically modifying weight updates based on performance feedback.

4.4.2. Learning Automata Integration

Learning automata improve training efficiency and prevent local minima:

- Monitors the error change after each epoch.
- If error decreases and remains below a threshold → reward current weights.
- If error increases or plateaus → penalize and replace weights.

Automata-based regulation enables:

- Faster convergence
- Prevention of overfitting
- Discarding ineffective weight configurations

4.4.3. Weight Update Mechanism

To optimize the neural network's weight assignment, a Learning Automaton (LA) is employed. Learning automata are decision-making systems modeled as abstract machines that operate in stochastic environments. These systems are capable of updating their decisions based on inputs received from the environment and corresponding probability values,

thereby improving overall system performance.

Each decision or action taken by the system is evaluated by the environment, which provides feedback to the learning automaton. Based on this feedback, the automaton updates its strategy and selects the most suitable action for the next step. A learning automaton can be represented as a triplet E={α, β, c}, where its components are defined as follows:

- Parameter α: This represents the set of inputs to the learning automaton, denoted as $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$. It corresponds to a set of actions or functions that may control a group of standard neurons, typically with a sigmoid activation function.
- Parameter β\beta: Represented as $\beta = \{\beta_1, \beta_2, \dots, \beta_n\}$, this parameter defines the set of outputs. It is a vector that describes the probability distribution over the actions selected by the Learning Control Unit (LCU).
- Parameter c: This feedback parameter, defined as c={0, 1}, represents the environment's response to the automaton's current action. A value of 0 indicates a penalty, while a value of 1 represents a reward.

Through iterative interaction with the environment, the learning automaton adjusts the probabilities of its actions to favor those that result in rewards, enabling it to effectively guide the neural network's learning process and improve performance.

a) Initial Data

In the proposed method, it is assumed that the neural network has N inputs. The neurons in the input layer act as a mapping function that transforms the N-dimensional input space into the required domains. Each neuron in the hidden layer represents a cluster of points related to the same group. The neurons in the output layer correspond to the number of classes.

During training, the network's weights and biases are adjusted so that the trained network responds to different inputs according to a specific rule.

It is assumed that there are K data samples, each having mm dimensions, where each dimension represents a feature in the proposed method. The dataset is categorized into two classes: intrusion and normal.

b) Weighting of Neural Network Layers

In each of these N functions, the network is initially trained with random weights, which are assumed to be identical for all functions at the start. The procedure is as follows: training data of the i^{th} class is fed into the first layer of the i^{th} function. The input data is denoted as x_i. These inputs are multiplied by their corresponding weights, which are initially assigned random values. The activation of each neuron in the hidden layer is then calculated according to Eq. (7), assuming v_{ij} as the weights and v_{oj} as the bias of the hidden layer.

$$Z_{in_j} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \quad (7)$$

The activation function in Eq. (8) is used to compute the output of the hidden (intermediate) neuron.

$$Z_j = f\left(Z_{in_j}\right) \quad (8)$$
$$F1(x) = \frac{1}{1+\exp(-x)}$$

The values obtained in the previous step are multiplied by the weights of the next layer, and the network output is then computed. Similar to the previous step, the activation function defined in Eq. (9) is used to obtain the output.

$$Y_{in_k} = w_{oj} + \sum_{j=1}^{p} z_i w_{jk} \quad (9)$$
$$Y_k = f\left(Y_{in_k}\right)$$

c) Evaluation of Assigned Weights

The error backpropagation for each output unit $Y_k = 1,2,\dots,m$, is computed as the difference between the network output and the actual output, defined by Eq. (10):

$$\delta_k = (t_k - y_k) f'(Y_{in_k}) \quad (10)$$

At this stage, a learning automaton is employed to check whether the propagation error falls below a predefined threshold. If the error is less than the threshold, this is considered a favorable condition for the network. Subsequently,

it is checked whether the current error is lower than the previous error. If so, it indicates that the network is progressing toward reducing the propagation error, which is desirable. As a partial reward, the network is allowed to continue its training. However, if the propagation error exceeds the previous epoch's error, the initially selected random weights are penalized. If this increase persists over several iterations, the penalization involves replacing the current weights with new random weights.

On the other hand, if the initial condition is not met — meaning the propagation error remains above the threshold and does not reduce below it over several predefined iterations — the corresponding weights are discarded and replaced with new weights. Finally, if the error falls below the lower threshold, the algorithm terminates.

This learning automaton mechanism offers several advantages: it prevents the learning error from escalating uncontrollably, eliminates inappropriate weights, and if certain weights produce a learning error exceeding the threshold, those weights are removed. These actions result in faster network training and, more importantly, yield accurate and appropriate weights that enhance the overall performance.

For each hidden unit (neuron) $z_j, j = 1,2, \dots, p$, the sum of the input deltas is computed as shown in Eq. (11):

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k \, w_{jk} \tag{11}$$

Using the delta values derived from the activation function, the formulas for updating the input layer weights connected to the hidden layer and the biases of the input layer are obtained as Eq. (12):

$$\Delta \, v_{ij} = \alpha v_{ij} = \alpha \delta_j x_j \tag{12}$$
$$\Delta \, v_{oj} = \alpha \delta_j$$

At this stage, based on the previously obtained information, the weights are updated. For all output units, weights and biases are updated according to Eq. (13), and for all hidden units, according to Equation (14):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \tag{13}$$

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \tag{14}$$

These steps are repeated for both classes in the proposed method. Through iteration of this process, the network is trained, producing a set of weight matrices corresponding to each class. For network testing and deployment, this procedure is repeated for each network.

5. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

5.1. Objective and Evaluation Criteria

The main objective of the evaluation is to assess the effectiveness and efficiency of the proposed hybrid intrusion detection system—based on Harris Hawks Optimization (HHO) for feature selection and neural network classification enhanced with learning automata—in detecting various types of cyber-attacks within IoT networks in smart cities.

The performance is assessed using standard evaluation metrics:

- Accuracy (overall correct classifications)
- True Positive Rate (TPR) (detection rate of actual attacks)
- False Positive Rate (FPR) (incorrect alarms on normal traffic)
- True Negative Rate (TNR) and False Negative Rate (FNR)

All metrics are calculated under both binary and multiclass classification scenarios using the KDDcup99 dataset, which includes five traffic types: Normal, DoS, Probe, R2L, U2R.

5.2. Experimental Setup

The proposed method was implemented using MATLAB 2017 and evaluated under simulated network conditions. All experiments were carried out on a system equipped with an Intel Core i7-10700 CPU running at 2.90 GHz and 16 GB of RAM. The software environment included Python 3.9, TensorFlow 2.x, and Scikit-learn libraries, operating on Ubuntu 22.04 LTS. Model performance was assessed using a 10-fold cross-validation approach to ensure robustness and reliability of the results. The simulation parameters are presented in Table 3.

Table 3. Simulation parameters

Parameter	Value
Simulation time	1500 seconds
Network size	100 × 100 m²
Number of nodes	100
HHO population	20 hawks
Maximum HHO iterations	100
Input features (after encoding)	122 features

5.3. Dataset Description

We use a preprocessed subset of the KDDCup99 dataset containing 494,021 labeled records. Each record consists of 41 features, categorized into five classes. The distribution of these classes along with the number of records per class is presented in Table 4.

Table 4. Class distribution and number of labeled records in the preprocessed KDDCup99 dataset used for model evaluation

Class Type	Description	Percentage (Train)	Percentage (Test)
Normal	Benign traffic	19.69%	19.48%
DoS	Denial of Service	79.24%	73.90%
Probe	Network reconnaissance	0.83%	1.34%
R2L	Remote access to local user	0.23%	5.21%
U2R	User privilege escalation	0.01%	0.07%

5.4. Performance Analysis

To comprehensively assess the performance of the proposed intrusion detection system, a series of evaluation charts was generated, each corresponding to a specific performance metric.

Fig. 2 illustrates the True Positive Rate (TPR) achieved by the proposed method for different categories of network attacks. The True Positive Rate (TPR) presents the system's ability to correctly detect various types of attacks. The highest TPR was observed for DoS attacks, reaching 99.21%, which is expected due to the high frequency and distinguishable patterns of such attacks. Probe attacks also achieved a strong TPR of 95.44%, reflecting the model's effectiveness in identifying scanning-based threats. More challenging attacks, such as R2L and U2R, which are typically underrepresented in the dataset and more difficult to detect, achieved TPRs of 93.86% and 91.13%, respectively. These results indicate that the proposed method performs well not only on frequent attacks but also on rare and subtle ones, thanks to the robust feature selection and adaptive classification components.

Fig. 3 illustrates the False Positive Rate (FPR) achieved by the proposed method for different categories of network attacks. The FPR illustrates the proportion of normal traffic mistakenly classified as malicious. The system demonstrated a low average FPR of 2.80%, with the lowest FPR recorded for DoS at 1.88%. These results are critical, as a high false alarm rate can burden security personnel and compromise the usability of a detection system. The low FPR across all classes highlights the model's capability to distinguish legitimate traffic from actual threats with high precision, ensuring operational efficiency in real-world deployments.

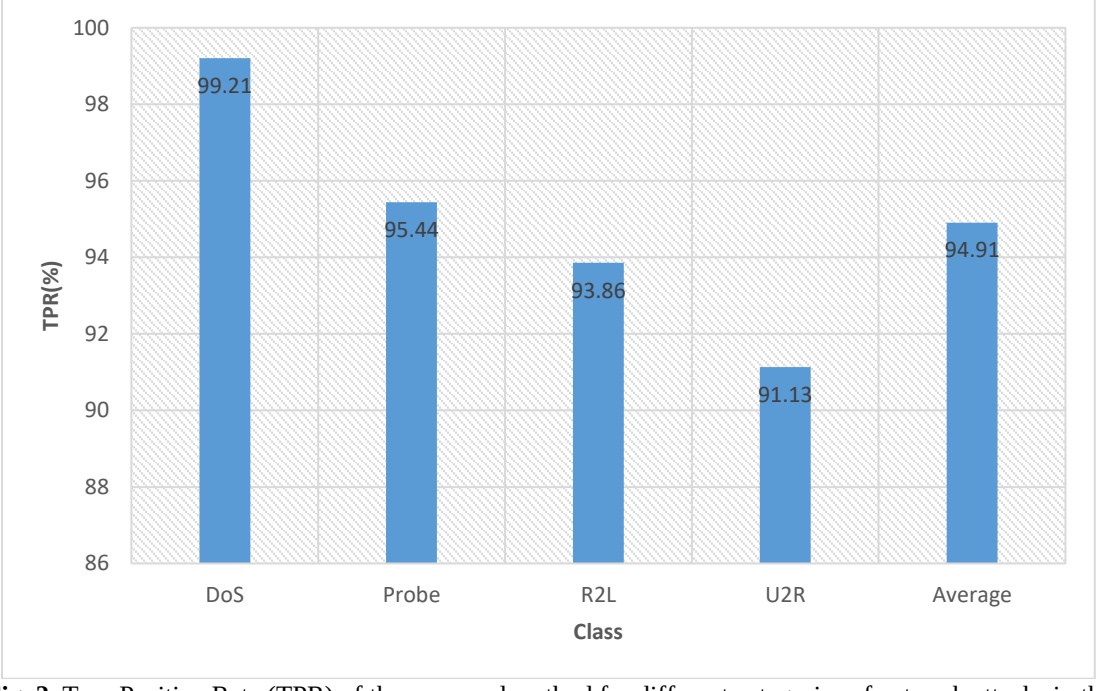

Fig. 2. True Positive Rate (TPR) of the proposed method for different categories of network attacks in the KDDCup99 dataset.

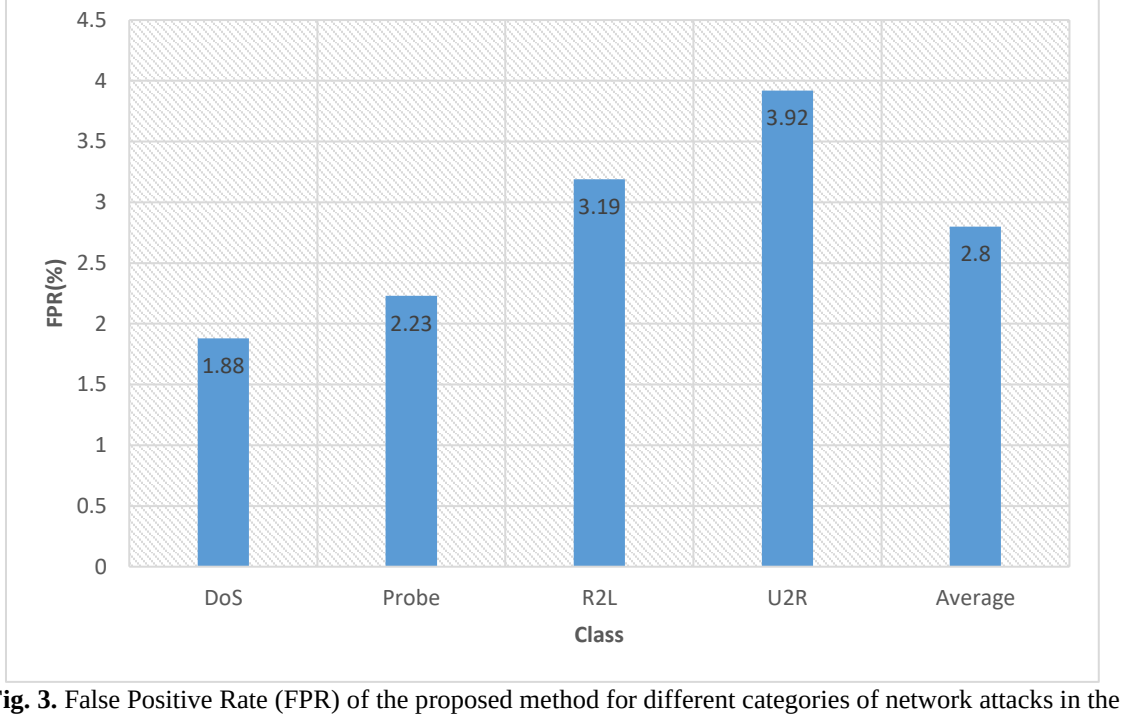

Fig. 3. False Positive Rate (FPR) of the proposed method for different categories of network attacks in the KDDCup99 dataset.

Fig. 4 illustrates the True Negative Rate (TNR) achieved by the proposed method for different categories of network attacks. TNR further confirms this capability by showing how accurately the system identifies normal, non-malicious traffic. The overall TNR reached 97.20%, with most classes (including Probe and R2L) exceeding 96%. Such high TNR values suggest that the model effectively minimizes false alarms and maintains reliability in classifying benign behavior—an essential requirement for smart city infrastructures where service continuity is critical.

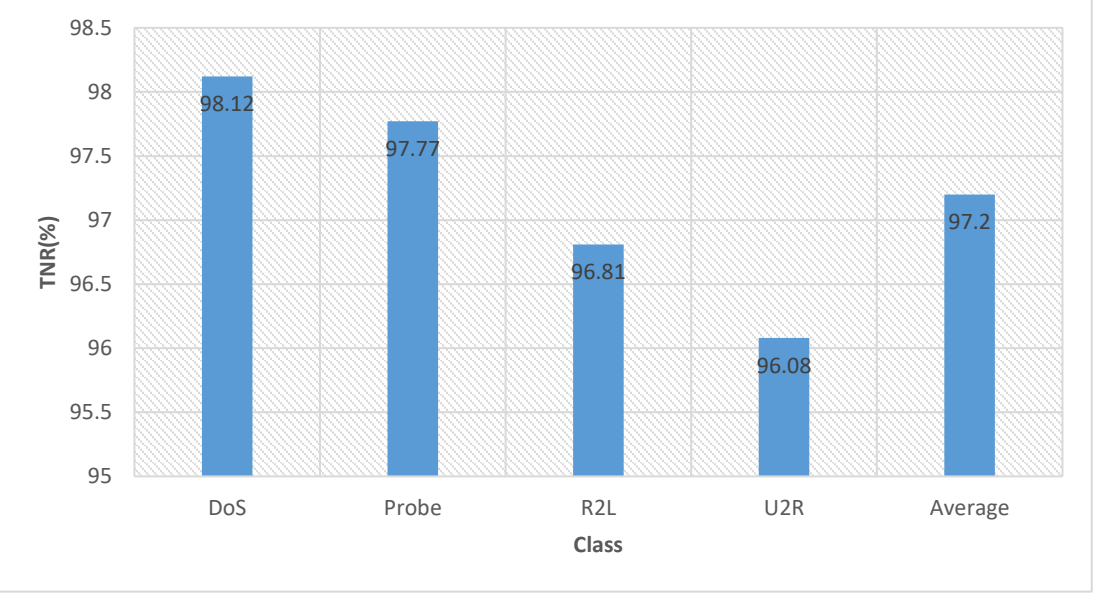

Fig. 4. True Negative Rate (TNR) of the proposed method for different categories of network attacks in the KDDCup99 dataset.

Fig. 5 illustrates the False Negative Rate (FNR) achieved by the proposed method for different categories of network attacks. On the other hand, the FNR quantifies the percentage of attacks that were incorrectly labeled as normal. The average FNR across all classes was 5.09%, with the lowest for DoS (0.79%) and the highest for U2R (8.87%). These values remain within acceptable thresholds for intelligent detection systems, especially given the difficulty of detecting rare attack types. A low FNR implies that the system is unlikely to miss actual threats, further enhancing its robustness and trustworthiness. The higher FNR observed for U2R (8.87%) is consistent with its rarity in the dataset. Despite this, the model outperforms many baseline approaches by achieving reliable detection of such low-frequency attack types.

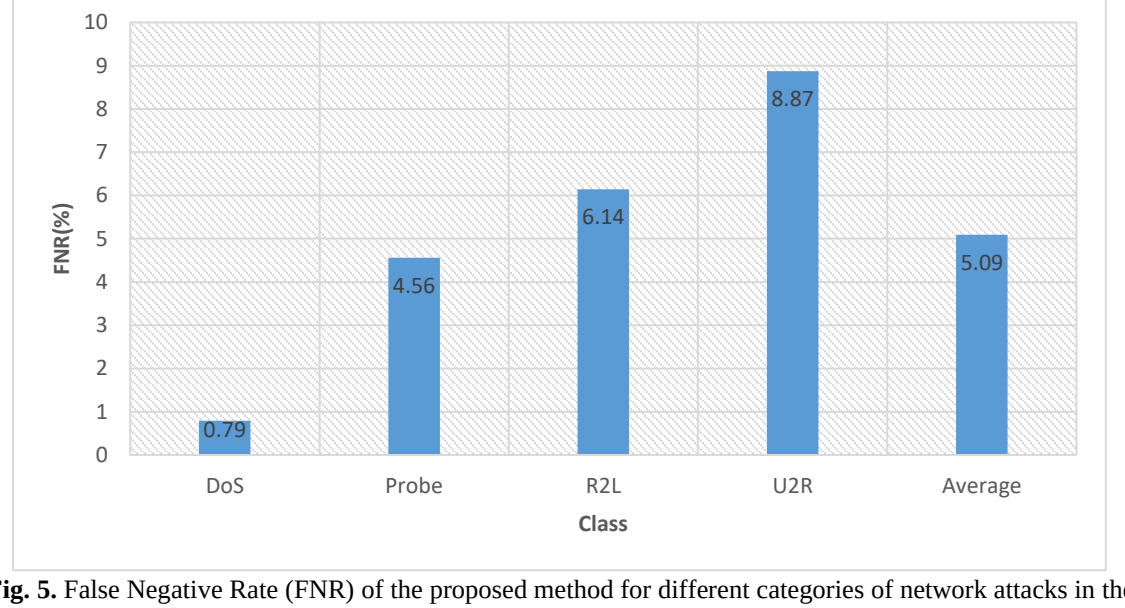

Fig. 5. False Negative Rate (FNR) of the proposed method for different categories of network attacks in the KDDCup99 dataset.

Fig. 6 illustrates the accuracy of the proposed method across different attack categories in the KDDCup99 dataset. The Accuracy shows the overall correctness of the system in classifying both attack and normal instances. The model achieved its highest accuracy for DoS attacks (98.67%) and its lowest for U2R (94.03%), with an average accuracy of 96.53% across all classes. This consistently high performance, even on difficult-to-detect classes, demonstrates the

effectiveness of integrating Harris Hawks Optimization for feature selection with learning automata-based neural network training, resulting in a well-generalized and stable classifier.

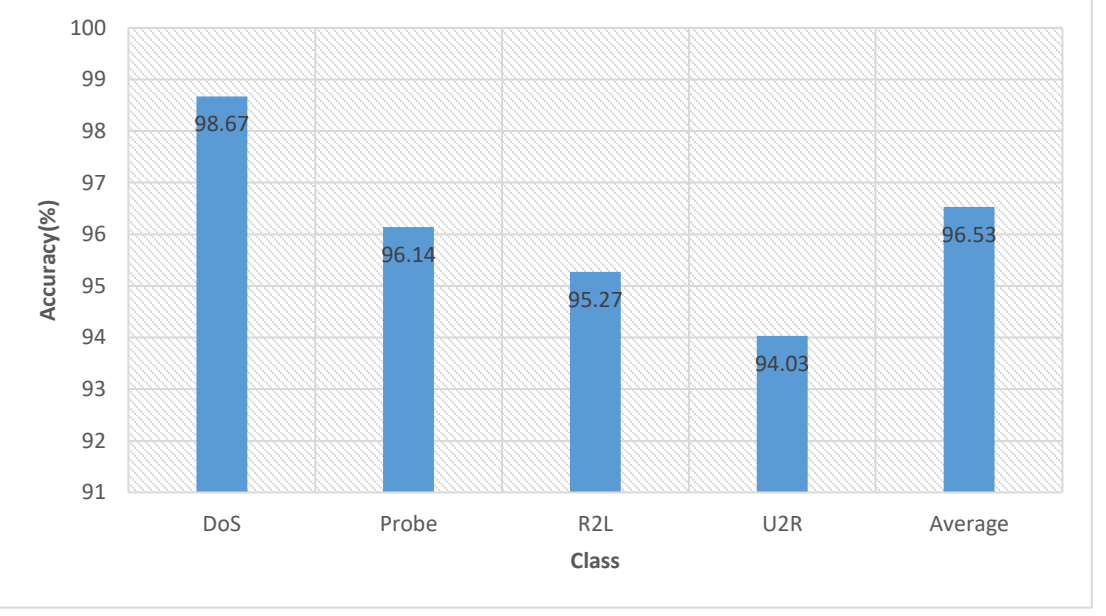

Fig. 6. Accuracy of the proposed method in detecting different categories of attacks using the KDDCup99 dataset.

Fig. 7 presents a comparative analysis of the proposed method against two recent approaches. In addition to individual class performance, a comparative chart was generated to benchmark the proposed model against two recent methods [13], [14]. The proposed method achieved an overall accuracy of 96.53%, compared to 94.32% and 95.18% for the respective reference methods. This performance gain of +2.2% and +1.35%, respectively, underscores the superiority of the proposed hybrid approach in both detection accuracy and system reliability. The improvement is mainly attributed to the dynamic and adaptive nature of the combined HHO and learning automata strategies, which enhance both training efficiency and classification accuracy.

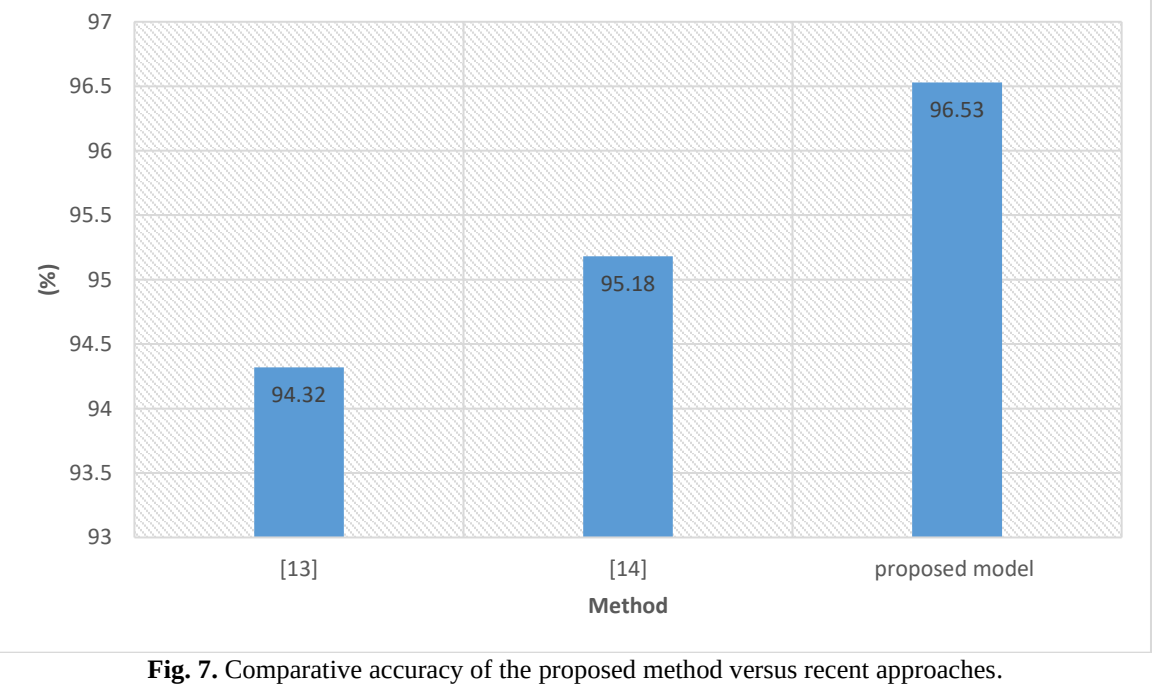

Fig. 7. Comparative accuracy of the proposed method versus recent approaches.

6. DISCUSSION

The proposed HHO + LA-NN model shows notable improvements in precision and recall across all classes, particularly for underrepresented attack types. The use of HHO reduces input dimensionality while preserving critical patterns, and Learning Automata improves the adaptiveness of the classifier. These combined mechanisms contribute to higher generalization, faster convergence, and lower false positive rates, making the system highly suitable for real-world IoT.

One limitation of our approach is the computational overhead introduced by the Harris Hawk Optimization (HHO) during feature selection, particularly for large-scale IoT datasets. Additionally, the Learning Automata mechanism requires fine-tuning of reward and penalty parameters, which may not generalize optimally across all IoT scenarios. Finally, our current evaluation is based on the KDDCup99 dataset, which, while widely used, may not fully capture the complexities of modern IoT traffic. Future work will address these limitations by experimenting with more realistic datasets and investigating online feature selection mechanisms.

For future work, we plan to extend our hybrid IDS in several directions. First, we aim to validate the system using more recent and realistic IoT datasets such as TON_IoT, CIC-IDS2017, and IoTID20, to better assess its applicability to real-world deployments. Second, we intend to develop an online variant of the model capable of real-time learning and adaptation to evolving attack patterns. Third, the proposed LA-NN architecture will be evaluated on embedded IoT hardware platforms (e.g., Raspberry Pi, Nvidia Jetson Nano) to analyze its performance under resource-constrained conditions. Finally, integration with edge and fog computing infrastructures will be explored to enable distributed and low-latency intrusion detection for smart city applications.

REFERENCES

[1] H. Allioui, and Y. Mourdi, "**Exploring the full potentials of IoT for better financial growth and stability: A comprehensive survey**," *Sensors*, Vol. 23, No. 19, pp. 8015, 2023.

[2] R., Ahmad, and I., Alsmadi, "**Machine learning approaches to IoT security: A systematic literature review,**" *Internet of Things*, Vol. 14, pp. 100365, 2021.

[3] W. H., Hassan, "**Current research on Internet of Things (IoT) security: A survey**," *Computer networks*, Vol. 148, pp. 283-294, 2019.

[4] K., He, D. D., Kim, M. R., and Asghar, "**Adversarial machine learning for network intrusion detection systems: A comprehensive survey**," *IEEE Communications Surveys & Tutorials*, Vol. 25, No. 1, pp. 538-566, 2023.

[5] N., Islam, F., Farhin, I., Sultana, M. S., Kaiser, M. S., Rahman, M. Mahmud, and G. H., Cho, "**Towards machine learning based intrusion detection in IoT networks**," *Comput. Mater*, Contin, Vol. 69, No. 2, pp. 1801-1821, 2021.

[6] M. A., Alsoufi, S., Razak, M. M., Siraj, I., Nafea, F. A., Ghaleb, F., Saeed, and M., Nasser, "**Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review**," *Applied sciences*, Vol. 11, No. 18, pp. 8383, 2021.

[7] A. N., Alsheavi, A., Hawbani, X., Wang, W., Othman, L., Zhao, Z., Liu, and M. A., Al-qaness, "**IoT Authentication Protocols: Classification, Trend and Opportunities**," *IEEE Transactions on Sustainable Computing*, 2024.

[8] A. A., Laghari, K., Wu, R. A., Laghari, M., Ali, and A. A., Khan, "**A review and state of art of Internet of Things (IoT)**," *Archives of Computational Methods in Engineering*, pp. 1-19, 2021.

[9] M. A., Khan, M. A., Khan, S. U., Jan, J., Ahmad, S. S., Jamal, A. A., Shah, and W. J., Buchanan, "A deep learning-based intrusion detection system for MQTT enabled IoT," *Sensors*, Vol. 21, No. 21, pp. 7016, 2021.

[10] R., Zhao, G., Gui, Z., Xue, J., Yin, T., Ohtsuki, B., Adebisi, and H., Gacanin, "**A novel intrusion detection method based on lightweight neural network for internet of things**," *IEEE Internet of Things Journal*, Vol. 9, No. 12, pp. 9960-9972, 2021.

[11] O. A., Wahab, "**Intrusion detection in the iot under data and concept drifts: Online deep learning approach**," *IEEE Internet of Things Journal*, Vol. 9, No. 20, pp. 19706-19716, 2022.

[12] M., Vishwakarma, and N., Kesswani, "**DIDS: A Deep Neural Network based real-time Intrusion detection system for IoT**," *Decision Analytics Journal*, Vol. 5, pp. 100142, 2022.

[13] S., Chatterjee, and M. K., Hanawal, "**Federated learning for intrusion detection in IoT security: a hybrid ensemble approach**," *International Journal of Internet of Things and Cyber-Assurance*, Vol. 2, No. 1, pp. 62-86, 2022.

[14] D. M., Ngo, D., Lightbody, A., Temko, C., Pham-Quoc, N. T., Tran, C. C., Murphy, and E., Popovici, E., "**HH-NIDS: Heterogeneous Hardware-Based Network Intrusion Detection Framework for IoT Security**," *Future Internet*, Vol. 15, No. 1, pp. 9, 2023.

[15] A., Awajan, "**A novel deep learning-based intrusion detection system for IOT networks**," *Computers*, Vol. 12, No. 2, pp. 34, 2023.

[16] Z., Wang, J., Li, S., Yang, X., Luo, D., Li, and S., Mahmoodi, "A lightweight IoT intrusion detection model based on improved BERT-of-Theseus," Expert Systems with Applications, Vol. 238, pp. 122045, 2024.

[17] A., Qaddos, M. U., Yaseen, A. S., Al-Shamayleh, M., Imran, A., Akhunzada, and S. Z., Alharthi, "**A novel intrusion detection framework for optimizing IoT security**," *Scientific Reports*, Vol. 14, No. 1, pp. 21789, 2024.

[18] N., Wang, S., Shi, Y., Chen, W., Lou, and Y. T., Hou, "**FeCo: Boosting intrusion detection capability in IoT networks via contrastive learning**," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[19] L., Lin, Q., Zhong, J., Qiu, and Z., Liang, "**E-GRACL: an IoT Intrusion Detection System Based on Graph Neural Networks**," *The Journal of Supercomputing*, Vol. 81, No. 1, pp. 42, 2025.
[20] M. A., Rahman, A. T., Asyhari, L. S., Leong, G. B., Satrya, M. H., Tao, and M. F., Zolkipli, "**Scalable machine learning-based intrusion detection system for IoT-enabled smart cities**," *Sustainable Cities and Society*, Vol. 61, pp. 102324, 2020.
[21] A., Alazab, M., Hobbs, J., Abawajy, and M., Alazab, "**Using feature selection for intrusion detection system**," *In 2012 international symposium on communications and information technologies (ISCIT*), IEEE, pp. 296-301, 2012.
[22] M., Tavallaee, E., Bagheri, W., Lu, and A. A., Ghorbani, "**A detailed analysis of the KDD CUP 99 data set**," *In 2009 IEEE symposium on computational intelligence for security and defense applications*, IEEE, pp. 1-6, 2009.
[23] A. A., Heidari, S., Mirjalili, H., Faris, I., Aljarah, M., Mafarja, and H., Chen, "**Harris hawks' optimization: Algorithm and applications**," *Future generation computer systems*, Vol. 97, pp. 849-872, 2019.
[24] K. T., Yang, "Artificial Neural Networks (ANNs): a New Paradigm for Thermal Science and Engineering", 2008.
[25] Thathachar, M. A., and P. S., Sastry, "Varieties of learning automata: an overview," *IEEE Transactions on Systems, Man, and Cybernetics*, Part B (Cybernetics), Vol. 32, No. 6, pp. 711-722, 2002.